

TECHNICAL DOCUMENT 3145
August 2002

Advanced Propagation Model (APM) Ver. 1.3.1 Computer Software Configuration Item (CSCI) Documents

A. E. Barrios
W. L. Patterson

Approved for public release;
distribution is unlimited



SSC San Diego
San Diego, CA 92152-5001

SSC SAN DIEGO
San Diego, California 92152-5001

T. V. Flynn, CAPT, USN
Commanding Officer

R. C. Kolb
Executive Director

ADMINISTRATIVE INFORMATION

The work detailed in this report was performed for Space and Naval Warfare Systems Command, PMW-155 by the Atmospheric Propagation Branch, Code 2858, of Space and Naval Warfare (SPAWAR) Systems Center, San Diego. Funding for this project was provided under program element 0603207N.

Released by
R. A. Paulus, Head
Atmospheric Propagation Branch

Under authority of
C. J. Sayre, Head
Electromagnetics and Advanced
Technology Division

This is a work of the United States Government and therefore is not copyrighted. This work may be copied and disseminated without restriction. Many SSC San Diego public release documents are available in electronic format at <http://www.spawar.navy.mil/sti/publications/pubs/index.html>

Contents

**SOFTWARE REQUIREMENTS SPECIFICATION
FOR THE ADVANCED PROPAGATION MODEL CSCI (Version 1.3.1)**

**SOFTWARE DESIGN DESCRIPTION
FOR THE ADVANCED PROPAGATION MODEL CSCI (Version 1.3.1)**

**SOFTWARE TEST DESCRIPTION
FOR THE ADVANCED PROPAGATION MODEL CSCI (Version 1.3.1)**

SOFTWARE REQUIREMENTS SPECIFICATION
FOR THE
ADVANCED PROPAGATION MODEL CSCI
(Version 1.3.1)

9 August 2002

Prepared for:
Space and Naval Warfare Systems Command (PMW-155)
San Diego, CA

Prepared by:
Space and Naval Warfare Systems Center, San Diego
Atmospheric Propagation Branch (Code 2858)
49170 Propagation Path
San Diego, CA 92152-7385

CONTENTS

1. SCOPE	1
1.1 Identification	1
1.2 System Overview	1
1.3 Document Overview	1
2. REFERENCED DOCUMENTS	1
3. REQUIREMENTS	4
3.1 CSCI Capability Requirements	4
3.1.1 Advance Propagation Model Initialization (APMINIT) CSC	9
3.1.1.1 Allocate Arrays APM (ALLARRAY_APM) SU	11
3.1.1.2 Allocate Array PE (ALLARRAY_PE) SU	11
3.1.1.3 Allocate Array RO (ALLARRAY_RO) SU	11
3.1.1.4 Allocate Array XORUF (ALLARRAY_XORUF) SU	11
3.1.1.5 Alpha Impedance Initialization (ALN_INIT) SU	11
3.1.1.6 Antenna Pattern (ANTPAT) SU	11
3.1.1.7 Dielectric Initialization (DIEINIT) SU	12
3.1.1.8 FFT Parameters (FFTPAR) SU	12
3.1.1.9 Fill Height Arrays (FILLHT) SU	12
3.1.1.10 Gaseous Absorption (GASABS) SU	12
3.1.1.11 Get Effective Earth Radius Factor (GET_K) SU	13
3.1.1.12 Get Alpha Impedance (GETALN) SU	13
3.1.1.13 Get Grazing Angle (GETGRAZE) SU	13
3.1.1.14 Get Maximum Angle (GETTHMAX) SU	13
3.1.1.15 Grazing Angle Interpolation (GRAZE_INT) SU	13
3.1.1.16 Interpolate Profile (INTPROF) SU	13
3.1.1.17 PE Initialization (PEINIT) SU	13
3.1.1.18 Profile Reference (PROFREF) SU	14
3.1.1.19 RD Trace (RDTRACE) SU	14
3.1.1.20 Refractivity Initialization (REFINIT) SU	14
3.1.1.21 Remove Duplicate Refractivity Levels (REMDUP) SU	14
3.1.1.22 Terrain Initialization (TERINIT) SU	15
3.1.1.23 Trace to Output Range (TRACE_ROUT) SU	15
3.1.1.24 Troposcatter Initialization (TROPOINT) SU	15
3.1.1.25 Starter Field Initialization (XYINIT) SU	15
3.1.2 Advanced Propagation Model Step (APMSTEP) CSC	15
3.1.2.1 Airborne Hybrid Model (AIRBORNE) SU	16
3.1.2.2 Calculate Propagation Loss (CALCLOS) SU	16
3.1.2.3 DoShift SU	17
3.1.2.4 Discrete Sine/Cosine Fast-Fourier Transform (DRST) SU	17
3.1.2.5 Flat-Earth Model (FEM) SU	17
3.1.2.6 Fast-Fourier Transform (FFT) SU	18
3.1.2.7 Free-Space Range Step (FRSTP) SU	18
3.1.2.8 FZLIM SU	18
3.1.2.9 Get Propagation Factor (GETPFAC) SU	18
3.1.2.10 Get Reflection Coefficient (GETREFCOEF) SU	19
3.1.2.11 Mixed Fourier Transform (MIXEDFT) SU	19
3.1.2.12 Parabolic Equation Step (PESTEP) SU	19
3.1.2.13 Ray Trace (RAYTRACE) SU	20
3.1.2.14 Refractivity Interpolation (REFINTER) SU	20
3.1.2.15 Ray Optics Calculation (ROCALC) SU	20
3.1.2.16 Ray Optics Loss (ROLOSS) SU	21

3.1.2.17 Save Profile (SAVEPRO) SU	21
3.1.2.18 Spectral Estimation (SPECEST) SU	22
3.1.2.19 Troposcatter (TROPOSCAT) SU	22
3.1.3 Extended Optics Initialization (XOINIT) CSC	22
3.1.3.1 Advanced Propagation Model Clean (APMCLEAN) SU	23
3.1.3.2 Mean Filter (MEANFILT) SU	23
3.1.4 Extended Optics Step (XOSTEP) CSC	23
3.1.4.1 Extended Optics (EXTO) SU	24
3.2 CSCI External Interface Requirements	24
3.3 CSCI Internal Interface Requirements	29
3.4 CSCI Internal Data Requirements	29
3.5 Adaptation Requirements	31
3.5.1 Environmental Radio Refractivity Field Data Elements	31
3.5.2 Terrain Profile Data Element	34
3.6 Security and Privacy Requirements	35
3.7 CSCI Environmental Requirements	35
3.8 Computer Resource Requirements	35
3.9 Software Quality Factors	35
3.10 Design and Implementation Constraints	36
3.10.1 Implementation and Application Considerations	36
3.10.2 Programming Language and Source Implementation	37
3.10.2.1 Programming Language	37
3.10.2.2 Source Implementation	38
3.11 Personnel-Related Requirements	39
3.12 Training-Related Requirements	39
3.13 Other Requirements	39
3.14 Precedence and Criticality of Requirements	40
 4. QUALIFICATION PROVISIONS	 40
 5. REQUIREMENTS TRACEABILITY	 40
5.1 System Traceability	40
5.2 Documentation Traceability	40
 6. NOTES	 48
 APPENDIX A	 51
A.1 Definitions of Quality Factor Criteria	51
A.2 Software Quality Metrics	52
A.2.1 Completeness Criteria	52
A.2.2 Consistency Criteria	52
A.2.3 Traceability Criteria	52

FIGURES

Figure 1. APM calculation regions.	4
Figure 2. Program flow of the APM CSCI.	6
Figure 3. APMINIT CSC program flow.	7
Figure 4. APMSTEP CSC program flow.	8
Figure 5. XOINIT CSC program flow.	8
Figure 6. XOSTEP CSC program flow.	9
Figure 7. Idealized M-unit profiles (solid) and lines of interpolation (dashed).	33

TABLES

Table 1. APM CSCI environmental data element requirements.	25
Table 2. APM CSCI external EM system data element requirements.	26
Table 3. APM CSCI external implementation data element requirements.	27
Table 4. APM CSCI external terrain data element requirements.	28
Table 5. APM CSCI output data element requirements.	29
Table 6. APMINIT SU returned error definitions.	30
Table 7. Requirements traceability matrix for the SRS.	42
Table 8. Acronyms and abbreviations.	49
Table 9. Fortran terms.	50

1. SCOPE

1.1 IDENTIFICATION

The Advanced Propagation Model (APM) Version 1.3.1 computer software configuration item (CSCI) calculates range-dependent electromagnetic (EM) system propagation loss within a heterogeneous atmospheric medium over variable terrain, where the radio-frequency index of refraction is allowed to vary both vertically and horizontally, also accounting for terrain and rough sea surface effects along the path of propagation.

1.2 SYSTEM OVERVIEW

The APM CSCI model will calculate propagation loss values as EM energy propagates through a laterally heterogeneous atmospheric medium where the index of refraction is allowed to vary both vertically and horizontally, also accounting for terrain effects along the path of propagation. Numerous Naval Integrated Tactical Environmental Subsystem (NITES) applications require EM-system propagation loss values. The required APM model described by this document may be applied to two such NITES applications, one of which displays propagation loss on a range versus height scale (commonly referred to as a coverage diagram) and one which displays propagation loss on a propagation loss versus range/height scale (commonly referred to as a loss diagram).

1.3 DOCUMENT OVERVIEW

This document specifies the functional requirements that are to be met by the APM CSCI. A discussion of the input software requirements is presented together with a general description of the internal structure of the APM CSCI as it relates to the CSCI's capability.

2. REFERENCED DOCUMENTS

1. Bergland, G. D., "A Radix-eight Fast Fourier Transform Subroutine for Real-valued Series," *IEEE Trans. Audio and Electro-acoust.*, Vol. AU-17, pp. 138-144, 1969.
2. Cooley, J. W., P. A. W. Lewis and P. D. Welsh, "The Fast Fourier Transform Algorithm: Programming Considerations in the Calculation of Sine, Cosine and Laplace Transforms," *J. Sound Vib.*, Vol. 12, pp. 315-337, 1970.

3. Tappert, F. D., "The Parabolic Approximation Method," Wave Propagation and Underwater Acoustics, J. B. Keller and J. S. Papadakis, Eds., New York, Springer-Verlag, pp. 224-285, 1977.
4. International Radio Consulting Committee (CCIR) XVth Plenary Assembly Dubrovnik, 1986, "Propagation in Non-Ionized Media," *Recommendations and Reports of the CCIR, 1986*, Vol. V, International Telecommunications Union, Geneva, 1986.
5. Commander-In-Chief, Pacific Fleet Meteorological Requirement (PAC MET) 87-04, "Range Dependent Electromagnetic Propagation Models," 1987.
6. Dockery, G. D., "Modeling Electromagnetic Wave Propagation in the Troposphere Using the Parabolic Equation," *IEEE Trans. On Antennas and Propagat.*, Vol. 36, No. 10, pp. 1464-1470, October 1988.
7. Naval Oceanographic Office, "Software Documentation Standards and Coding Requirements for Environmental System Product Development," April 1990.
8. Kuttler, J. R. and G. D. Dockery., "Theoretical Description of the Parabolic Approximation/Fourier Split-Step Method of Representing Electromagnetic Propagation in the Troposphere," *Radio Sci.*, Vol. 26, pp. 381-393, 1991.
9. Dockery, G. D. and J. R. Kuttler. "An Improved Impedance-Boundary Algorithm for Fourier Split-Step Solutions of the Parabolic Wave Equation," *IEEE Trans. On Antennas and Propagat.*, Vol. 44, No. 12, pp. 1592-1599 December 1996
10. American National Standards Institute (ANSI), "Program Language Fortran Extended," 1992.
11. Patterson, W.L. and H. V. Hitney. "Radio Physical Optics CSCI Software Documents," Naval Command, Control and Ocean Surveillance Center, RDT&E Division, San Diego, CA, TD 2403, December 1992.
12. Barrios, A. E., "Terrain and Refractivity Effects on Non-Optical Paths," AGARD Conference Proceedings 543, Multiple Mechanism Propagation Paths (MMPPs): Their Characteristics and Influence on System Design, pp. 10-1 to 10-9, October 1993.
13. Barrios, A. E., "A Terrain Parabolic Equation Model for Propagation in the Troposphere," *IEEE Trans. Antennas Propagat.*, Vol. 42, pp. 90-98, January 1994.

14. Naval Oceanographic Office, "Software Documentation Standards for Environmental System Product Development," February 1996.
15. Barrios, A. E., "Terrain Parabolic Equation Model (TPEM) Version 1.5 User's Manual," Naval Command, Control and Ocean Surveillance Center, RDT&E Division, San Diego, CA, NRaD TD 2898, February 1996.
16. Sailors, D. B. and A. E. Barrios. "Terrain Parabolic Equation Model (TPEM) Computer Software Configuration Item (CSCI) Documents," Naval Command, Control and Ocean Surveillance Center, RDT&E Division, San Diego, CA, TD 2963, May 1997.
17. Sailors, D. B., A. E. Barrios, W. L. Patterson, and H. V. Hitney. "Advanced Propagation Model [Ver. 1.0] (APM) Computer Software Configuration Item (CSCI) Documents," SSC San Diego, San Diego, CA, TD 3033, August 1998.
18. Horst, M. M., F. B. Dyer, and M. T. Tuley. "Radar Sea Clutter Model," IEEE International Conference on Antennas and Propagation.
19. Miller, A. R., R. M. Brown, and E. Vegh, "New Derivation for the Rough-Surface Reflection Coefficient and for the Distribution of Sea-Wave Elevations" *Proc. IEEE*, Vol. 131, Part H, 2, pp 114-116, 1984.

3. REQUIREMENTS

3.1 CSCI CAPABILITY REQUIREMENTS

The required APM CSCI propagation model is a range-dependent true hybrid model that uses the complementary strengths of both Ray Optics (RO) and Parabolic Equation (PE) techniques to calculate propagation loss both in range and altitude.

The atmospheric volume is divided into regions which lend themselves to the application of the various propagation loss calculation methods. Figure 1 illustrates these regions.

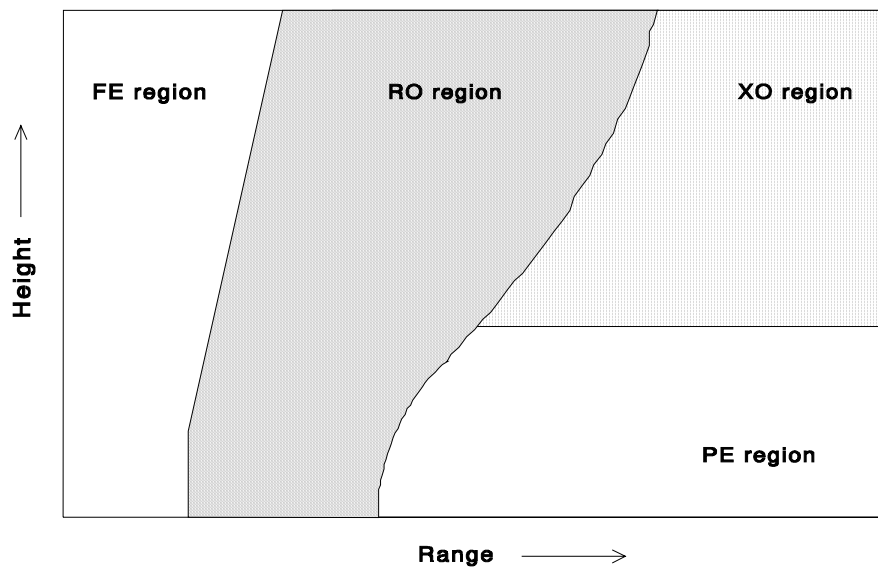


Figure 1. APM calculation regions.

For antenna elevation angles above 5 degrees or for ranges less than approximately 2.5 kilometers (km), a flat-earth (FE) ray-optics model is used. In this region, only receiver height is corrected for average refraction and earth curvature.

Within the RO region (as defined by a limiting ray), propagation loss is calculated from the mutual interference between the direct-path and surface-reflected ray components using the refractivity profile at zero range. Full account is given to focusing or de-focusing along both direct and reflected ray paths and to the integrated optical path length difference between the two ray paths, to give precise phase difference, and, hence, accurate coherent sums for the computation of propagation loss.

For the low-altitude region beyond the RO region, a PE approximation to the Helmholtz full wave equation is employed. The PE model allows for range-dependent refractivity profiles and variable terrain along the propagation path and uses a split step Fourier method for the solution of the PE. The PE model is run in the minimum region required to contain all terrain and trapping layer heights.

For the area beyond the RO region but above the PE region, an extended optics region (XO) is defined. Within the XO region, ray-optics methods which are initialized by the PE solution given below are used.

APM will run in three “execution” modes depending on environmental inputs. APM will use the FE, RO, XO, and PE models if the terrain profile is flat for the first 2.5 km and if the antenna height is less than or equal to 100 m. It will use only the XO and PE models if the terrain profile is *not* flat for the first 2.5 km and if the antenna height is less than or equal to 100 m. For applications in which the antenna height is greater than 100 meters, a combination of FE and PE methods are used. The FE model is used for all propagation angles greater than $\pm 5^\circ$ from the source and the PE model is used for angles less than $\pm 5^\circ$. By default, APM will automatically choose which mode of operation it will use for a specified set of inputs. However, the ability to run only the PE model for any case is allowed by setting a logical flag upon input.

The APM CSCI allows for horizontal and vertical antenna polarization, finite conductivity based on user-specified ground composition and dielectric parameters, rough sea surface effects, and the complete range of EM system parameters and most antenna patterns required by NITES. APM also allows for gaseous absorption effects in all sub-models and computes troposcatter losses within the diffraction region and beyond.

The program flow of the required APM CSCI is illustrated Figure 2. Note that the APM CSCI is shown within the context of a calling CSCI application such as one that generates a coverage or loss diagram. The efficient implementation of the APM CSCI will have far reaching consequences upon the design of an application CSCI beyond those mentioned in Section 3.10. For example, Figure 2 shows checking for the existence of a previously created APM output file prior to the access of the APM CSCI. The application CSCI will have to consider if the atmospheric or terrain environment has changed since the APM output file was created or if any new height or range requirement is accommodated within the existing APM CSCI output file. Because these and many more considerations are beyond the scope of this document to describe, an application CSCI designer should work closely with the APM CSCI development agency in the implementation of the APM CSCI. Figure 3 through Figure 6 illustrate the program flow for the main compute software components (CSC), APMINIT CSC, APMSTEP CSC, XOINIT CSC, and XOSTEP CSC, respectively.

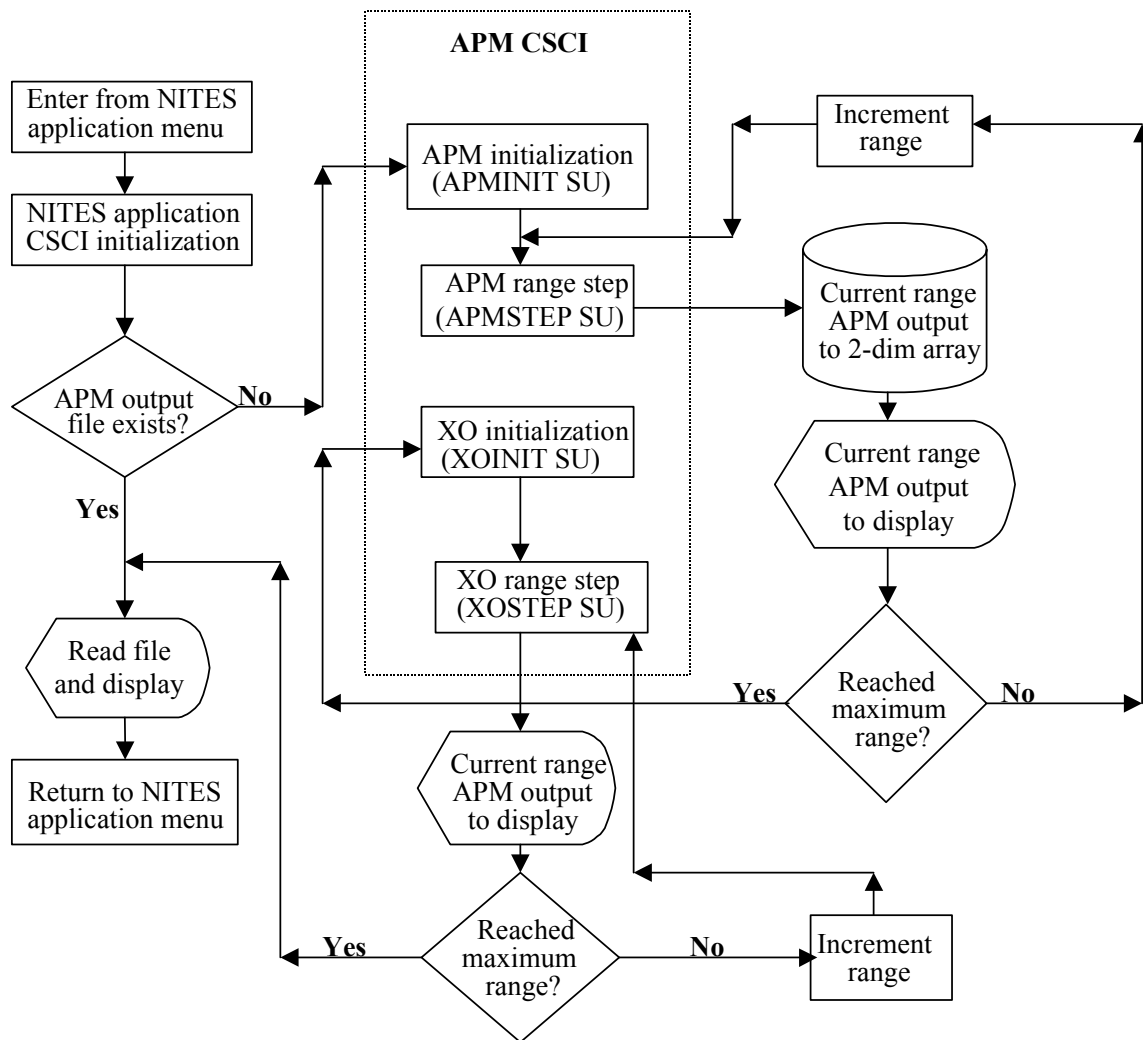


Figure 2. Program flow of the APM CSCI.

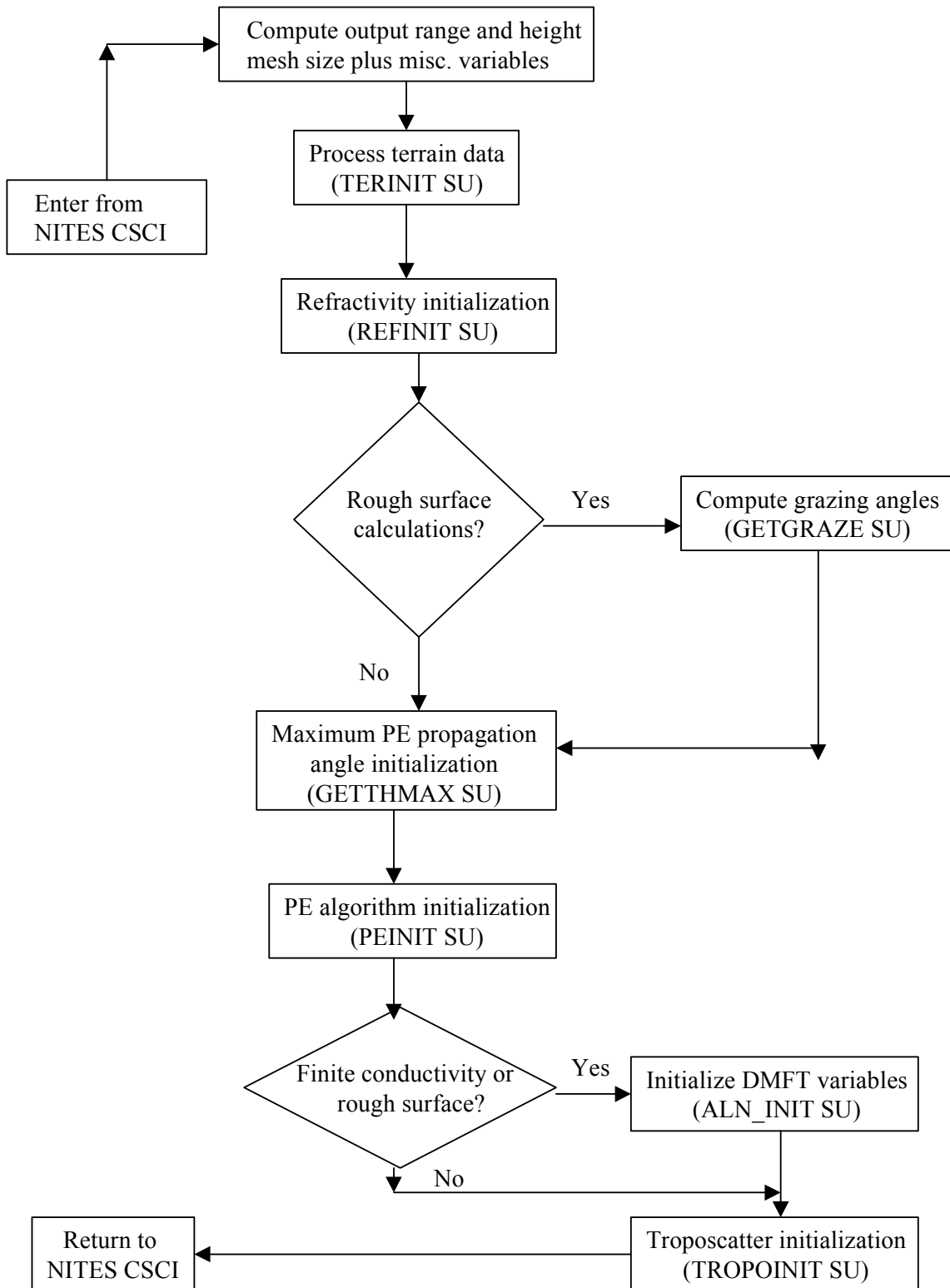


Figure 3. APMINIT CSC program flow.

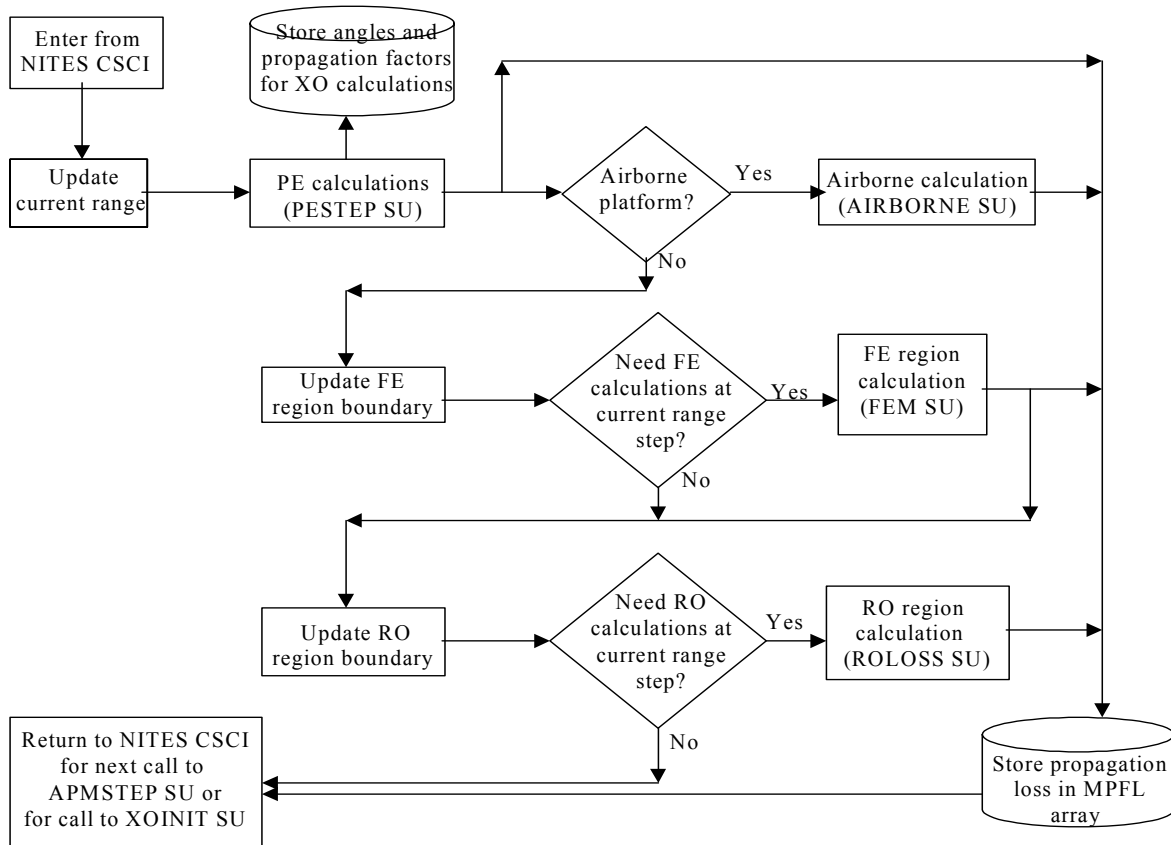


Figure 4. APMSTEP CSC program flow.

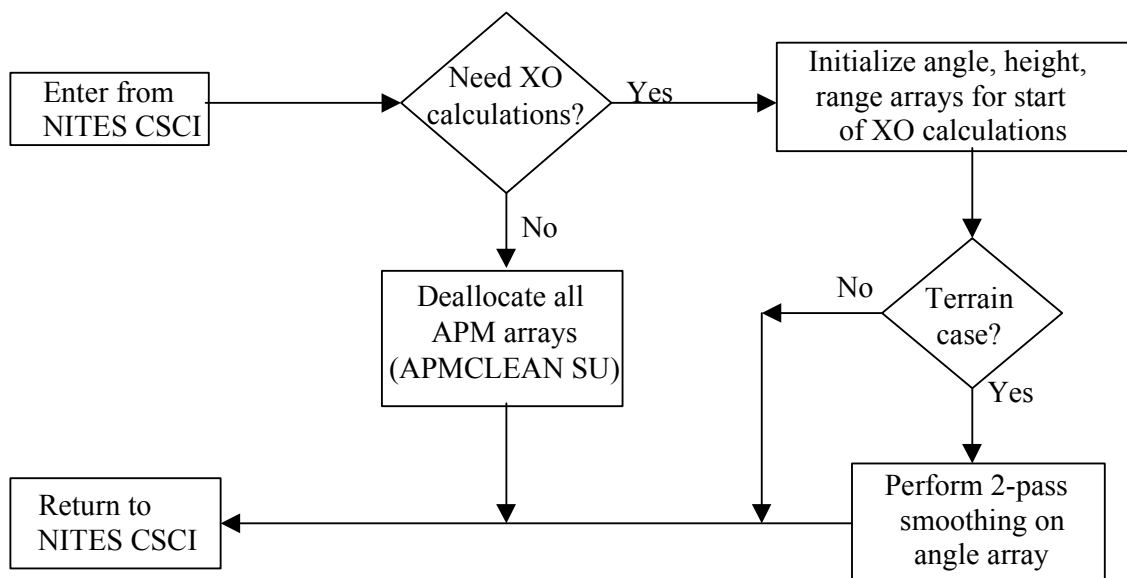


Figure 5. XOINIT CSC program flow.

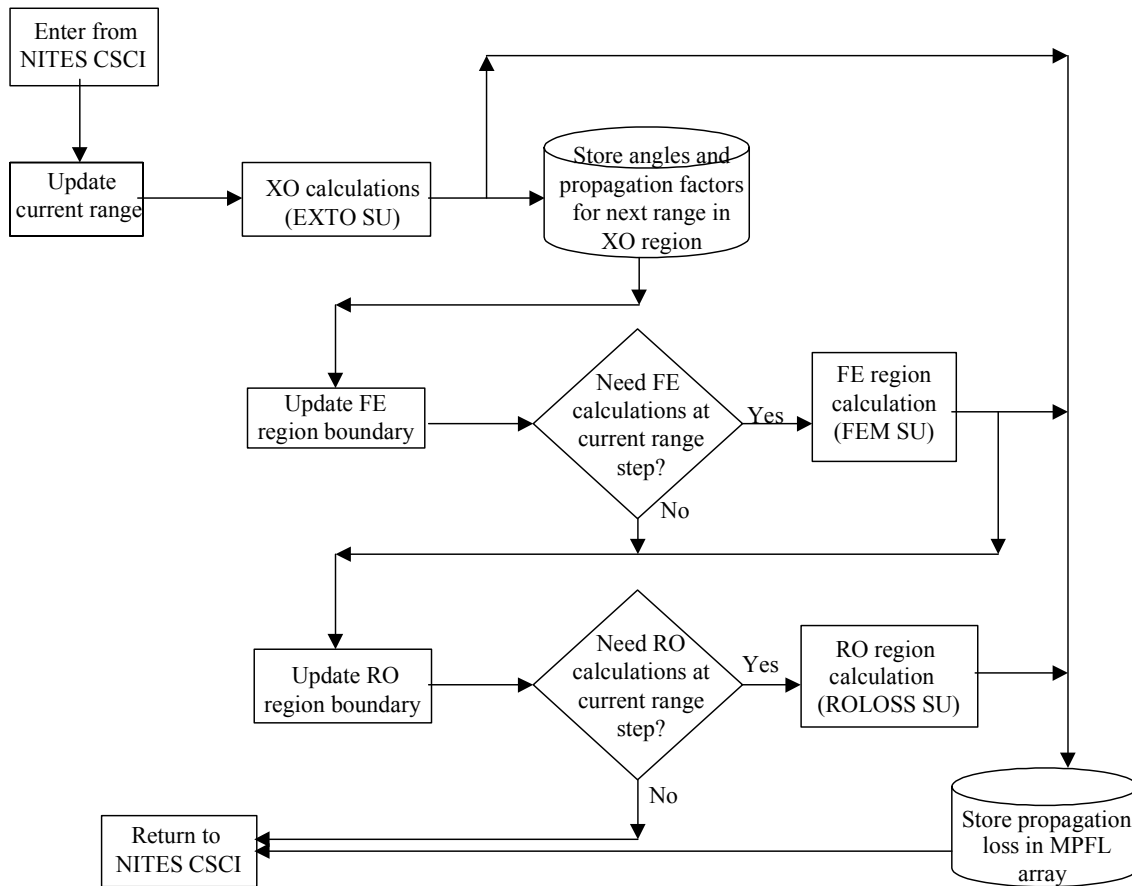


Figure 6. XOSTEP CSC program flow.

The APM CSCI is divided into four main computer software components (CSC) and 47 additional software units (SUs). The name, purpose, and a general description of processing required for each SU follows.

3.1.1 Advanced Propagation Model Initialization (APMINIT) CSC

The purpose of the APMINIT CSC is to interface with various SUs for the complete initialization of the APM CSCI.

The atmospheric volume must be “covered” or resolved with a mesh of calculation points which will normally exceed the height/range resolution requirements of the particular application of the APM CSCI. Upon entering the APMINIT CSC, a range and height grid size per the APM CSCI output point is calculated from the number of APM outputs and the maximum CSCI range and height.

A TERINIT SU is referenced to examine the terrain profile. The minimum terrain height is determined, and then the entire terrain profile is adjusted by this height so that

all internal calculations are referenced to this height. This is done in order to minimize the required PE transform size.

User-specified environmental and system parameters are examined to determine if the APM CSCI will execute in a full hybrid mode, a partial hybrid mode, or PE-only mode.

A REFINIT SU is referenced to initialize the NITES CSCI specified modified refractivity and also to test for valid environment profiles. A PROFREF SU adjusts the environment profiles by the internal reference height, and a INTPROF SU defines the modified refractivity at all PE vertical mesh points.

If rough surface calculations are required (i.e., a non-zero wind speed is specified) a GETGRAZE SU is referenced to determine the grazing angles for the given refractivity profile. The grazing angles are then sorted and a GRAZE_INT SU is referenced to combine grazing angles computed by both methods (if necessary) and interpolate for subsequent use in rough surface calculations.

In order to automatically determine the maximum PE calculation angle, a GETTHMAX SU is referenced. This determines, via ray tracing, the minimum angle for which adequate coverage can be given with the specified terrain and environment profile. A FFTPAR SU is referenced to determine the Fast-Fourier transform (FFT) size for the calculated angle and to initialize data elements within the PE region which are dependent on the size of the FFT. The minimum size for the FFT is determined from the Nyquist criterion. There is also an option to activate *only* the PE algorithm within the APM CSCI, regardless of inputs. If this option is enabled, the PE maximum calculation angle is supplied by the calling CSCI.

If vertical polarization is specified, then additional calculations are performed in the starter solution using Kuttler and Dockery's mixed transform method (Refs. 8, 9). In this case, a DIEINIT SU is used to initialize dielectric ground constants. For general ground types, the permittivity and conductivity are calculated as a function of frequency from curve fits to the permittivity and conductivity graphs shown in recommendations and reports of the International Radio Consulting Committee (Ref. 4).

A PE initialization SU (PEINIT) is referenced to initialize all variables and arrays associated with PE calculations. A XYINIT SU and an antenna pattern factor SU (ANTPAT) are referenced to generate a first solution to the PE. A FFT SU is referenced for data elements required in obtaining the PE's starting solution.

If rough surface calculations are required, or if performing a finite-conducting boundary case, then a ALN_INIT SU is referenced to initialize field variables used in the mixed transform algorithm.

If running in a full hybrid mode, a FILLHT SU is referenced to determine the heights at each output range separating the FE, RO, and PE calculation regions. If running in a partial hybrid or PE-only mode, then the heights at each output range are determined, below which propagation loss/factor solutions are valid. No propagation loss/factor solutions are provided above these heights for those execution modes.

A TROPOINT SU is referenced to initialize all variables and arrays associated with troposcatter calculations. This is referenced only if the user has enabled the T_{ropo} flag.

Finally, a GASABS SU is referenced to initialize the attenuation parameter due to oxygen and water vapor absorption.

3.1.1.1 Allocate Arrays APM (ALLARRAY_APM) SU

The purpose of the ALLARRAY_APM SU is to allocate and initialize all dynamically dimensioned arrays associated with APM terrain, refractivity, troposcatter, and general variable arrays.

3.1.1.2 Allocate Array PE (ALLARRAY_PE) SU

The purpose of the ALLARRAY_PE SU is to allocate and initialize all dynamically dimensioned arrays associated with PE calculations.

3.1.1.3 Allocate Array RO (ALLARRAY_RO) SU

The purpose of the ALLARRAY_RO SU is to allocate and initialize all dynamically dimensioned arrays associated with RO calculations.

3.1.1.4 Allocate Array XORUF (ALLARRAY_XORUF) SU

The purpose of the ALLARRAY_XORUF SU is to allocate and initialize all dynamically dimensioned arrays associated with XO and rough surface calculations.

3.1.1.5 Alpha Impedance Initialization (ALN_INIT) SU

The purpose of the ALN_INIT SU is to initialize variables used in the discrete mixed Fourier transform (DMFT) algorithm for finite conductivity and/or rough surface calculations.

3.1.1.6 Antenna Pattern (ANTPAT) SU

The purpose of the ANTPAT SU is to calculate a normalized antenna gain (antenna pattern factor) for a specified antenna elevation angle.

From the antenna beam width, elevation angle (an angle for which the antenna pattern factor is desired) and the antenna radiation pattern type, an antenna factor is calculated.

3.1.1.7 Dielectric Initialization (DIEINIT) SU

The purpose of the DIEINIT SU is to determine the conductivity and relative permittivity as functions of frequency in MHz based on general ground composition types.

3.1.1.8 FFT Parameters (FFTPAR) SU

The purpose of the FFTPAR SU is to determine the required transform size based on the maximum PE propagation angle and the maximum height needed. If running in full or partial hybrid modes, the maximum height needed is the height necessary to encompass at least 20% above the maximum terrain peak along the path or the highest trapping layer specified in the environment profiles, whichever is greater. If running in a PE-only mode, the maximum height needed is the specified maximum output height. An error is returned if the computed transform size reaches 2^{30} .

For computational efficiency reasons, an artificial upper boundary must be established for the PE solution. To prevent upward propagating energy from being “reflected” downward from this boundary and contaminating the PE solution, the PE solution field strength should be attenuated or “filtered” above a certain height to ensure that the field strength just below this boundary is reduced to zero.

The total number of vertical points for which a transformation will be computed is determined. This term is also referred to as the FFT size. The filtering boundary height is also determined.

3.1.1.9 Fill Height Arrays (FILLHT) SU

The purpose of the FILLHT SU is to calculate the effective earth radius for an initial launch angle of 5° and to fill an array with height values at each output range of the limiting sub-model, depending on which mode is being used. If running in a full hybrid mode, the array contains height values at each output range separating the FE from the RO region. If running in other modes, the array contains those height values at each output range at which the initial launch angle has been traced to the ground or surface. For an execution mode in which PE and XO models are used, these height values represent the separating region where, above that height, valid loss is computed, and below that height, no loss is computed. This is done so only loss values that fall within a valid calculation region are output. For airborne applications in which a combination of PE and FE models are used, the array contains the height values at each range separating the FE and PE regions.

3.1.1.10 Gaseous Absorption (GASABS) SU

The purpose of the GASABS SU is to compute the specific attenuation based on air temperature and absolute humidity. This SU is based on CCIR (International Telecommunication Union, International Radio Consultative Committee, now the ITU-R) Recommendation 676-1, “Attenuation by Atmospheric Gases in the Frequency Range 1-350 GHz.”

3.1.1.11 Get Effective Earth Radius Factor (GET_K) SU

The purpose of the GET_K SU is to compute the effective earth radius factor and the effective earth radius. The computation is made for a launch angle of 5° if the SU is called from the APMINIT CSC. If called from the TROPOINIT SU, then the computation is made for a launch angle equal to the critical angle.

3.1.1.12 Get Alpha Impedance (GETALN) SU

The purpose of the GETALN SU is to compute the impedance term in the Leontovich boundary condition, and the complex index of refraction for finite conductivity, vertical polarization, and rough sea surface calculations. These formulas follow Kuttler and Dockery's method (Ref. 9).

3.1.1.13 Get Grazing Angle (GETGRAZE) SU

The purpose of the GETGRAZE SU is to compute the grazing angles at each PE range step for subsequent use in rough sea surface calculations. The grazing angle computation is done using two methods. The first method uses ray trace by referencing a RDTRACE SU to compute grazing angles. The second is done by spectral estimation of the near-surface PE field. The near-surface PE field is computed by performing a "PE run" for a smooth surface, perfect conducting, horizontal polarization case at a fixed frequency (10 GHz) and fixed PE maximum calculation angle (4°).

3.1.1.14 Get Maximum Angle (GETTHMAX) SU

The purpose of the GETTHMAX SU is to perform an iterative ray trace to determine the minimum angle required (based on the reflected ray) in obtaining a PE solution. The determination of this angle will depend on the particular mode of execution. For the full and partial hybrid modes, a ray is traced up to a height that exceeds at least 20% above the maximum terrain peak along the path or the highest trapping layer specified in the environment profiles, whichever is greater. The maximum PE propagation angle is then determined from the local maximum angle of the traced ray.

3.1.1.15 Grazing Angle Interpolation (GRAZE_INT) SU

The purpose of the GRAZE_INT SU is to interpolate grazing angles at each PE range step based on angles computed from ray trace (takes precedence) and those computed from spectral estimation. These are used for subsequent rough surface calculations.

3.1.1.16 Interpolate Profile (INTPROF) SU

The purpose of the INTPROF SU is to perform a linear interpolation vertically with height on the refractivity profile. Interpolation is performed at each PE mesh height point.

3.1.1.17 PE Initialization (PEINIT) SU

The purpose of the PEINIT SU is to initialize all variables used in the PE model for subsequent calls to the PESTEP SU. The PE calculation height mesh and range step

size are determined. The free-space propagator term is computed at each PE angle, or p-space, mesh point using the wide-angle propagator. A filter, or attenuation function (frequently called “window”), is applied to the upper $\frac{1}{4}$ of the array corresponding to the highest $\frac{1}{4}$ of the maximum propagation angle. Next, the environmental phase term is computed at each PE height, or z-space, mesh point (if performing a smooth surface, refractive homogeneous case). A filter, or attenuation function (frequently called “window”), is applied to the upper $\frac{1}{4}$ of the mesh points corresponding to the highest $\frac{1}{4}$ of the calculation height domain. Finally, a XYINIT SU is referenced to initialize the PE starting field.

3.1.1.18 Profile Reference (PROFREF) SU

The purpose of the PROFREF SU is to adjust the current refractivity profile so that it is relative to a reference height. The reference height is initially the minimum height of the terrain profile. Upon subsequent calls from the PESTEP SU, the refractivity profile is adjusted by the local ground height at each PE range step.

3.1.1.19 RD Trace (RDTRACE) SU

The purpose of the RDTRACE SU is to perform ray traces of many rays launched within an angle of $\pm 4^\circ$. All angles from rays striking the surface are then sorted and stored for subsequent interpolation in the GRAZE_INT SU.

3.1.1.20 Refractivity Initialization (REFINIT) SU

The purpose of the REFINIT SU is to check for valid environmental profile inputs and to initialize all refractivity arrays.

The environmental data is checked for a range-dependent profile and tested to determine if the range of the last profile entered is less than the maximum output range specified. If so, an integer error flag is returned and the SU exited, depending on the values of logical error flags set in the NITES CSCI itself.

The REFINIT SU also tests for valid refractivity level entries for each profile. If the last gradient in any profile is negative, it returns an integer error flag and the SU is exited. If no errors are detected, the REFINIT SU then extrapolates the environmental profiles vertically to 1000 km in height. Extrapolation is not performed horizontally from the last provided profile; rather, the last provided environment profile is duplicated at 10^7 km in range. This duplication of profiles is done by the REFINTER SU. Finally, the REFINIT SU checks if an evaporation duct profile has been specified.

3.1.1.21 Remove Duplicate Refractivity Levels (REMDUP) SU

The purpose of the REMDUP SU is to remove any duplicate refractivity levels in the currently interpolated profile.

3.1.1.22 Terrain Initialization (TERINIT) SU

The purpose of the TERINIT SU is to examine and initialize terrain arrays for subsequent use in PE calculations. It tests for and determines a range increment if it is found that range/height points are provided in fixed range increments. The minimum terrain height is determined and the entire terrain profile is adjusted so all internal calculations are referenced to this height. This is done in order to maximize the PE transform calculation volume.

3.1.1.23 Trace to Output Range (TRACE_ROUT) SU

The purpose of the TRACE_ROUT SU is to trace a single ray, whose launch angle is specified by the calling routine, to each output range. The height of this ray is stored at each output range for subsequent proper indexing and accessing of the appropriate sub-models.

3.1.1.24 Troposcatter Initialization (TROPOINT) SU

The purpose of the TROPOINT SU is to initialize all variables and arrays needed for subsequent troposcatter calculations. The tangent range and tangent angle are determined from the source and from all receiver heights and stored in arrays.

3.1.1.25 Starter Field Initialization (XYINIT) SU

The purpose of the XYINIT SU is to calculate the complex PE solution at range zero.

Several constant terms which will be employed over the entire PE mesh are calculated. These are the angle difference between mesh points in p-space and a height-gain value at the source (transmitter).

For each point in the PE p-space mesh, the antenna pattern ANTPAT SU is referenced to obtain an antenna pattern factor for both a direct-path ray and a surface-reflected ray. The complex portions of the PE solution are then determined from the antenna pattern factors, elevation angle, and antenna height. The initial field assumes the source is over a perfectly conducting ground.

3.1.2 Advanced Propagation Model Step (APMSTEP) CSC

The purpose of the APMSTEP CSC is to advance the entire APM CSCI algorithm one output range step, referencing various SUs to calculate the propagation loss at the current output range. At this current range, APM calculations will be made within the vertical (up to the maximum PE height region) by accessing the appropriate region's SUs.

The current output range is determined. The PESTEP SU is referenced to obtain the PE portion of the propagation loss at this new range.

For an airborne application, the AIRBORNE SU is referenced to obtain the FE portion of the propagation loss above and below the PE maximum propagation angle.

If running in full hybrid mode, then based upon a height array index used within the FE region, a determination is made for the necessity to include FE propagation calculations. If so, the FEM SU is referenced to obtain the FE portion of the propagation loss. If a FE calculation is made, the maximum height index for the RO region is adjusted (with the minimum height index corresponding to the maximum height index of the PE region), and the ROM SU is referenced to obtain the RO portion of the propagation loss at the current range. FE and RO propagation loss will be computed only up to the range at which XO calculations will be performed.

If running in a partial hybrid (PE + XO) mode, then only the PESTEP SU will be referenced to obtain the PE portion of the propagation loss at this new range.

Finally, absorption loss is computed for the current range and added to the propagation loss at all heights.

3.1.2.1 Airborne Hybrid Model (AIRBORNE) SU

The purpose of the AIRBORNE SU is to determine propagation loss based on flat-earth calculations for the direct ray path only for regions above and below the PE maximum propagation angle.

3.1.2.2 Calculate Propagation Loss (CALCLOS) SU

The purpose of the CALCLOS SU is to determine the propagation loss from the complex PE field at each output height point at the current output range.

The local ground height at the current output range is determined. All propagation loss/factor values at output height points up to the local ground height are then set to -999. The first valid loss point is determined corresponding to the first output height point above the ground height. Next, the last valid loss point is determined based on the smaller of the maximum output height or the height traced along the maximum PE propagation angle to the current output range.

From the height of the first valid loss point to the height of the last valid loss point, the GETPFAC SU is referenced to obtain the propagation factor in dB (field strength relative to free space) at all corresponding output heights at the previous and current PE ranges. Then, for each valid output height, horizontal interpolation in range is performed to obtain the propagation factor at the current output range. From the propagation factor and the free-space loss, the propagation loss at each valid output height is determined, with the propagation loss/factor set to -1000 for all output height points above the last valid output height but less than the maximum output height.

If running in full or partial hybrid modes, the propagation factor at the top of the PE region is determined at every output range and stored in an array for future reference in XO calculations. If troposcatter calculations are desired, the TROPOSCAT SU is referenced with the results added to the propagation loss/factor array. Absorption loss is computed for the current range and added to the propagation loss at all heights. All loss/factor values returned to the NITES CSCI at this point are in centibels ($10 \text{ cB} = 1 \text{ dB}$).

3.1.2.3 DoShift SU

The purpose of the DOSHIFT SU is to shift the complex PE field by the number of bins, or PE mesh heights corresponding to the local ground height.

Upon entering, the number of bins to be shifted are determined. The PE solution is then shifted downward if the local ground is currently at a positive slope, and upward if the local ground is at a negative slope.

3.1.2.4 Discrete Sine/Cosine Fast-Fourier Transform (DRST) SU

A function with a common period, such as a solution to the wave equation, may be represented by a series consisting of sines and cosines. This representation is known as a Fourier series. An analytical transformation of this function, known as a Fourier transform, may be used to obtain a solution for the function.

The solution to the PE approximation to Maxwell's wave equation is to be obtained by using such a Fourier transformation function. The APM CSCI requires the real-valued sine and cosine transformations in which the real and imaginary parts of the PE equation are transformed separately. A Fourier transformation for possible use with the APM CSCI is described by Bergland (Ref. 1) and Cooley, Lewis, and Welsh (Ref. 2).

3.1.2.5 Flat-Earth Model (FEM) SU

The purpose of the FEM SU is to compute propagation loss at a specified range based upon flat-earth approximations. Receiver heights are corrected for earth curvature and average refraction based on twice the effective earth radius computed in the GET_K SU. The following steps are performed for each APM output height.

1. The path lengths and elevation angles for both the direct-path and surface-reflected path, along with the grazing angle, are computed from simple right triangle calculations. Using the two elevation angles, the ANTPAT SU is referenced to obtain an antenna pattern factor for each angle. Using the grazing angle, the GETREFCOEF SU is referenced to obtain the magnitude and phase lag of the surface reflection coefficient.
2. From the path length difference, the phase lag of the surface reflected ray, and the wave number, a total phase lag is determined. Using the total phase lag, the magnitude of the surface reflection coefficient and the two antenna pattern factors, the

two ray components are coherently summed to obtain a propagation factor. The propagation factor, together with the free-space propagation loss and path length difference of the direct-path ray are used to compute the propagation loss.

3.1.2.6 Fast-Fourier Transform (FFT) SU

The purpose of the FFT SU is to separate the real and imaginary components of the complex PE field into two real arrays and then reference the DRST SU which transforms each portion of the PE solution.

3.1.2.7 Free-Space Range Step (FRSTP) SU

The purpose of the FRSTP SU is to propagate the complex PE solution field in free space by one range step.

The PE field is transformed to p-space and then multiplied by the free-space propagator. Before exiting, the PE field is transformed back to z-space. Both transforms are performed using a FFT SU.

3.1.2.8 FZLIM SU

The purpose of the FZLIM SU is to determine both the propagation factor (in dB) and the outgoing propagation angle at the top of the PE calculation region. These values, along with the corresponding PE range, are stored for future reference by the XOINIT SU.

The GETPFAC SU is referenced to determine the propagation factor at the last height mesh point in the valid part of the PE region. The propagation factor, along with the range and the local ray angle (determined from the ray traced separating the RO and PE regions), is stored if this is the first call to the FZLIM SU. The SPECEST SU is then referenced to determine the outgoing propagation angle. Depending on the change of angles from one range step to the next, the calculated outgoing angle will be limited. The storage array counter is incremented and the outgoing angle stored.

Before exiting, the SAVEPRO SU is referenced to store the refractivity profiles from the top of the PE region to the maximum specified coverage height.

3.1.2.9 Get Propagation Factor (GETPFAC) SU

The purpose of the GETPFAC SU is to determine the propagation factor at the specified height in dB.

A vertical interpolation with height on the PE solution field is performed to obtain the magnitude of the field at the desired output height point. An additional calculation is made and the propagation factor is then returned in dB.

3.1.2.10 Get Reflection Coefficient (GETREFCOEF) SU

The purpose of the GETREFCOEF SU is to calculate the complex surface reflection coefficient, along with the magnitude and phase angle.

The complex reflection coefficient is computed from a specified grazing angle and is based on the Fresnel reflection coefficient equations for vertical and horizontal polarization. The magnitude and phase angle are determined from the complex reflection coefficient. If rough surface calculations are required, the smooth surface reflection coefficient is then modified by the Miller-Brown rough surface reduction factor, which is a function of wind speed and grazing angle.

3.1.2.11 Mixed Fourier Transform (MIXEDFT) SU

The purpose of the MIXEDFT SU is to propagate the PE field in free space one PE range step, applying the Leontovich boundary condition, using the mixed Fourier transform as outlined by Kuttler and Dockery (Ref. 9). For finite conducting boundaries (i.e., if vertical polarization is specified or rough surface calculations are required) and the frequency is less than 400 MHz, the central difference form of the DMFT is used. If the frequency is greater than 400 MHz, the backward difference form of the DMFT is used.

3.1.2.12 Parabolic Equation Step (PESTEP) SU

The purpose of the PESTEP SU is to advance the PE solution one output range step, referencing various SUs to calculate the propagation loss at the current output range.

The next output range is determined and an iterative loop begun to advance the PE solution such that for the current PE range, a PE solution is calculated from the solution at the previous PE range. This procedure is repeated until the output range is reached.

At each PE range step, the local ground height is determined and the PE field is “shifted” by the number of bins, or PE mesh height points, corresponding to the local ground height. This is performed in the DOSHIFT SU.

If the APM CSCI is being used in a range-dependent mode, that is, more than one profile has been input, or a terrain profile is specified, the REFINTER SU is referenced to compute a new modified refractive index profile adjusted by the local ground height at the current range. A new environmental phase term is computed using this new refractivity profile.

If using vertical polarization and the current ground type has changed from the previous one, or rough surface calculations are required, a GETALN SU is referenced to determine the impedance term and all associated variables used for the mixed transform calculations.

If rough surface calculations are required, or if using vertical polarization, then a MIXEDFT SU is referenced; otherwise, a FRSTP SU is referenced to advance the PE solution one range step in free space. The environmental phase term is then applied to obtain the new final PE solution at the current range. Once all calculations are made to determine the PE field at the current PE range, the FZLIM SU is referenced to determine and store the outgoing propagation factor and propagation angle at the top of the PE region. The FZLIM SU is only referenced if running in full or partial hybrid modes. Finally, a CALCLOS SU is referenced to obtain the propagation loss/factor at the desired output heights at the current output range.

3.1.2.13 Ray Trace (RAYTRACE) SU

Using standard ray trace techniques, a ray is traced from a starting height and range with a specified starting elevation angle to a termination range. As the ray is being traced, an optical path length difference and a derivative of range with respect to elevation angle are being continuously computed. If the ray should reflect from the surface, a grazing angle is determined. Upon reaching the termination range, a terminal elevation angle is determined along with a termination height.

3.1.2.14 Refractivity Interpolation (REFINTER) SU

The purpose of the REFINTER SU is to interpolate both horizontally and vertically on the modified refractivity profiles. Profiles are then adjusted so they are relative to the local ground height.

If range-dependent refractive profiles have been specified, horizontal interpolation to the current PE range is performed between the two neighboring profiles. A REMDUP SU is referenced to remove duplicate refractivity levels, and the PROFREF SU is then referenced to adjust the new profile relative to the internal reference height corresponding to the minimum height of the terrain profile. The PROFREF SU is referenced once more to adjust the profile relative to the local ground height, and upon exit from the PROFREF SU, the INTPROF SU is referenced to interpolate vertically on the refractivity profile at each PE mesh height point.

3.1.2.15 Ray Optics Calculation (ROCALC SU)

The purpose of the ROCALC SU is to compute the RO components needed in the calculation of propagation loss at a specified range and height within the RO region. These components are the amplitudes for a direct-path and surface-reflected ray, and the total phase lag angle between the direct-path and surface-reflected rays.

A test is made to determine if this is the first RO calculation. If an initial calculation is needed, the height, range, and elevation angle array indices are set to initial conditions. If not, the array indices are incremented from the previous RO calculation.

The following steps are performed for each series of vertical grid points, in a manner that ensures that RO calculations have been performed at ranges that span the

current range of interest. The vertical grid points are taken in order beginning with the one with greatest height.

1. Using a Newton iteration method with a varying elevation angle, the RAYTRACE SU is referenced to find a direct-path ray and a surface-reflected ray originating at the transmitter height and terminating at the same grid point. Should a direct or reflected ray not be found to satisfy the condition, or should the computed grazing angle exceed the grazing angle limit, the height array index is adjusted to redefine the lower boundary of the RO region. Should the ray trace conditions be satisfied, the RAYTRACE SU will provide a terminal elevation angle, a derivative of range with respect to elevation angle, a path length, and for the surface-reflected ray, a grazing angle.
2. Using the final direct-path ray and surface-reflected ray elevation angles obtained from the Newton iteration method, the ANTPAT SU is referenced to obtain an antenna pattern factor for each angle. The GETREFCOEF SU is referenced to obtain the amplitude and phase lag angle of the surface reflection coefficient.
3. Using the antenna pattern factors, path length differences, and surface-reflection coefficients, the necessary RO components defined in the first paragraph above are calculated.

3.1.2.16 Ray Optics Loss (ROLOSS) SU

The purpose of the ROLOSS SU is to calculate propagation factor and loss values at all valid RO heights at a specified range based upon the components of magnitude for a direct-path and surface-reflected ray and the total phase lag angle between the two rays as determined by the ROCALC SU.

For purposes of computational efficiency, an interpolation from the magnitude and total phase lag arrays, established by the ROCALC SU, is made to obtain these three quantities at each APM vertical output mesh point within the RO region.

From the interpolated phase lag and ray amplitudes, a propagation factor is calculated which is used, in turn, with the free-space loss to obtain a propagation loss at each vertical APM output point. Absorption loss is computed for the current range and added to the propagation loss at all heights.

3.1.2.17 Save Profile (SAVEPRO) SU

The purpose of the SAVEPRO SU is to store refractivity profiles at each PE range step from the top of the PE region to the maximum user-specified height. This is only done if running in full or partial hybrid modes.

The refractivity height level just exceeding the PE region height limit is determined. From this level upward, all heights, M-units, and gradients are stored.

3.1.2.18 Spectral Estimation (SPECEST) SU

The purpose of the SPECEST SU is to determine the outward propagation angle at the top of the PE calculation region or the grazing angle at the lower part of the PE region based on spectral estimation. The outward propagation angle is used for XO calculations and the grazing angle is used for rough surface calculations.

The upper 8 (if running smooth surface case) or 16 (if running terrain case) bins of the complex PE field at the current PE range are separated into their real and imaginary components. The upper $\frac{1}{4}$ of this portion of the field is then filtered and zero-padded to 256 points. It is then transformed to its spectral components via a reference to the DRST SU. The amplitudes of the spectral field are then determined and a 3-point average is performed. The peak of the 256-point field is found and the outgoing propagation angle is determined from the peak value.

3.1.2.19 Troposcatter (TROPOSCAT) SU

The purpose of the TROPOSCAT SU is to determine the loss due to troposcatter and to compute the appropriate loss from troposcatter and propagation loss for large receiver ranges.

The current output range is updated and the tangent angle from the source to the current output range is initialized. For all output receiver heights at the current output range, the following procedure is performed.

1. If the current output range is less than the minimum diffraction field range for a particular receiver height, then the SU is exited and no troposcatter loss is computed.
2. The tangent angle from the receiver height is determined.
3. The common volume scattering angle is determined and calculations are performed to obtain the loss due to troposcatter.
4. Troposcatter loss is compared to propagation loss. If the difference between the propagation loss and troposcatter loss is less than 18 dB, then the corresponding power levels of the two loss values are added. If the difference is greater than 18 dB, then the lesser of the two losses is used.

3.1.3 Extended Optics Initialization (XOINIT) CSC

The purpose of the XOINIT SU is to initialize the range, height, and angle arrays in preparation for the XOSTEP CSC.

Upon entering, if XO calculations are not required, the APMCLEAR SU is referenced to deallocate all arrays used for the current application, then the CSC is exited.

If XO calculations are required, all arrays used for XO calculations are allocated and initialized to 0. The ranges and angles previously stored from referencing the FZLIM SU are now used to initialize the range and angle arrays for XO calculations. A 10-point smoothing average on the angle array is performed twice via reference to the MEANFILT SU. Upon exiting, the height array and initial height index for start of XO calculations are initialized.

3.1.3.1 Advanced Propagation Model Clean (APMCLEAR) SU

The purpose of the APMCLEAR CSC is to deallocate all dynamically dimensioned arrays used in one complete run of APM calculations.

3.1.3.2 Mean Filter (MEANFILT) SU

The purpose of the MEANFILT SU is to perform an n-point average smoothing on any array passed to it.

3.1.4 Extended Optics Step (XOSTEP) CSC

The purpose of the XOSTEP CSC is to advance the APM CSCI algorithm one output range step from the top of the PE calculation region to the maximum output height specified, referencing various SUs to calculate the propagation loss and factor at the current output range.

Upon entering the XOSTEP CSC, the current output range is determined. The EXTO SU is referenced to obtain the XO portion of the propagation loss and factor at this new range.

If running in full hybrid mode, based upon a height array index used within the FE region, determine if it's necessary to include FE propagation calculations. If necessary, the FEM SU is referenced to obtain the FE portion of the propagation loss and factor. If a FE calculation is made, the maximum height index for the RO region is adjusted (with the minimum height index corresponding to the maximum height index of the PE region), and the ROLOSS SU is referenced to obtain the RO portion of the propagation loss and factor at the current range.

If running in partial hybrid mode, then only the EXTO SU is referenced to obtain the XO portion of the propagation loss and factor at this new range. The maximum height will correspond to the maximum user-specified coverage height.

Finally, the APMCLEAR SU is referenced to deallocate all allocated arrays used throughout the run.

3.1.4.1 Extended Optics (EXTO) SU

The purpose of the EXTO SU is to calculate propagation loss and factor, based on extended optics techniques, at the current output range.

Upon entering, array indices for the current range, height, and angle arrays are initialized. A ray trace is then performed for all rays from the last output range to the current output range. The current heights are then sorted, along with their corresponding propagation factors. The propagation loss is then determined at each output receiver height by interpolation on the terminal heights of the traced rays.

Upon exiting, a reference to the TROPOSCAT SU provides any troposcatter losses and is added to the loss array. Absorption loss is also added to the propagation loss at all heights.

3.2 CSCI EXTERNAL INTERFACE REQUIREMENTS

The APM CSCI is accessed, through the APMINIT CSC, by a subroutine call from the NITES CSCI which should provide, as global data elements, the values specified in Table 1 through Table 4.

The APM CSCI external data elements, i.e., those data which must be provided by the calling NITES CSCI prior to the APM CSCI execution may be divided into four classifications. The first is external data related to the atmospheric environment, specified within Table 1; the second is data related to the EM system, specified within Table 2; the third is data related to the implementation of the APM CSCI by the NITES CSCI, specified within Table 3; and the fourth is data related to the terrain information, specified within Table 4. Each table lists the type, units, and bounds of each data element. Table 5 specifies the output data of the APM CSCI model.

Table 1. APM CSCI environmental data element requirements.

Name	Description	Type	Units	Bounds
<i>refmsl</i>	Modified refractivity profile (dynamically allocated) array referenced to mean sea level	real	M	$\geq 0.0^a$
<i>hmsl</i>	Profile height (dynamically allocated) array	real	meters	See note b
<i>n_{prof}</i>	Number of refractivity profiles	integer	N/A	≥ 1
<i>lvlp</i>	Number of profile levels	integer	N/A	≥ 2
<i>rngprof</i>	Dynamically allocated array of ranges to each profile	real	meters	≥ 0.0
<i>abs_{hum}</i>	Surface absolute humidity	real	g/m ³	0 to 50 ^c
<i>t_{air}</i>	Surface air temperature	real	°C	-20 to 40 ^c
<i>γ_a</i>	Surface specific attenuation	real	dB/km	≥ 0.0
<i>i_{extra}</i>	Extrapolation flag for refractivity profiles entered in combination with terrain below mean sea level	integer	N/A	0 or 1
<i>n_w</i>	Number of wind speeds and corresponding ranges	integer	N/A	≥ 0.0
<i>rngwind</i>	Dynamically allocated array of ranges specified for each wind speed in <i>wind()</i> .	real	meters	≥ 0.0
<i>wind</i>	Dynamically allocated array of wind speeds.	real	meters/second	0.0 to 10.0

^aCouplets of height and modified refractivity associated with that height are referred to within this document as a refractivity profile.

^bAll heights in the refractivity profile must be steadily increasing.

^cThe CCIR gaseous absorption model implemented within APM provides a $\pm 15\%$ accuracy for absolute humidity and surface air temperature within these bounds. While values beyond these limits are allowed within APM, it should be noted this may result in less accurate attenuation rates calculated.

Table 2. APM CSCI external EM system data element requirements.

Name	Description	Type	Units	Bounds
μ_{bw}	Antenna vertical beam width	real	degree	.5 to 45
μ_o	Antenna elevation angle	real	degree	-10.0 to 10.0
f_{MHz}	EM system frequency	real	MHz	100.0 to 20,000.0
i_{pat}	Antenna pattern 1 = Omni-directional 2 = Gaussian 3 = Sine (X)/X 4 = Cosecant-squared 5 = Generic height-finder 6 = User-defined height-finder 7 = User-defined antenna pattern	integer	N/A	1 to 7
i_{pol}	Antenna polarization 0 = Horizontal 1 = Vertical	integer	N/A	0 to 1
ant_{ht}	Antenna height above local ground at range 0.0 m	real	meters	$\geq 1.5^a$
$hfang$	Dynamically allocated user-defined height-finder power reduction angle array ($i_{pat}=6$) or antenna pattern angles ($i_{pat}=7$)	real	degree	0.0 to 90.0 for $i_{pat}=6$ -90.0 to 90.0 for $i_{pat}=7$
$hffac$	Dynamically allocated user-defined power reduction factor array ($i_{pat}=6$) or antenna pattern factors ($i_{pat}=7$)	real	N/A	0.0 to 1.0
n_{facs}	Number of power reduction angles/factors for user-defined height-finder antenna pattern	integer	N/A	1 to 10

^aThe minimum antenna height will vary depending on the frequency and beamwidth according to the formula:

$$ant_{ht} \geq \text{maximum of} \left(1.5, 6 \frac{c_o}{f_{mhz} \mu_{bw}} \right)$$

where c_o is the speed of light x 10^{-6} m/s (299.79245).

Table 3. APM CSCI external implementation data element requirements.

Name	Description	Type	Units	Bounds
h_{max}	Maximum height output for a particular application of APM	real	meters	$\geq 100.0^a$
h_{min}	Minimum height output for a particular application of APM	real	meters	$\geq 0.0^a$
$lerr6$	Logical flag to allow for error -6 to be bypassed	logical	N/A	'true.' or 'false.' ^b
$lerr12$	Logical flag to allow for error -12 to be bypassed	logical	N/A	'true.' or 'false.' ^b
n_{rout}	Number of range output points for a particular application of APM	integer	N/A	≥ 1
n_{zout}	Number of height output points for a particular application of APM	integer	N/A	≥ 1
PE_{flag}	Logical flag to enable only the PE model for a particular application of APM	logical	N/A	'true.' or 'false.' ^c
r_{max}	Maximum range output for a particular application of APM	real	meters	$\geq 5000.0^b$
r_{mult}	PE-range step multiplier	real	N/A	$> 0.0^c$
th_{max}	Visible portion of PE maximum calculation angle	real	degrees	$> 0.0^c$
T_{ropo}	Logical flag to include troposcatter calculations.	integer	N/A	'true.' or 'false.'

^a refer to section 3.5.2 for a complete description.

^b refer to section 3.5.1 for a complete description.

^c refer to section 3.4 for a complete description.

Table 4. APM CSCI external terrain data element requirements.

Name	Description	Type	Units	Bounds
<i>terx</i>	Dynamically allocated terrain profile range array	real	meters	$\geq 0.0^a$
<i>tery</i>	Dynamically allocated terrain profile height array	real	meters	$\geq 0.0^a$
<i>i_{tp}</i>	Number of terrain profile points for a particular application of APM	integer	N/A	≥ 2
<i>i_{gr}</i>	Number of ground types for a particular application of APM	integer	N/A	$\geq 0.0^a$
<i>igrnd</i>	Array of ground composition types for a particular application of APM 0 = Sea water 1 = Fresh water 2 = Wet ground 3 = Medium dry ground 4 = Very dry ground 5 = Ice at -1° C 6 = Ice at -10° C 7 = User defined	integer	N/A	$0 \leq igrnd \leq 7^a$
<i>rgrnd</i>	Dynamically allocated array of ranges for which ground types are applied for a particular application of APM	real	meters	$\geq 0.0^a$
<i>dielec</i>	Dynamically allocated 2-dimensional array of relative permittivity (ϵ_r) and conductivity (σ) for a particular application of APM	real	ϵ_r - N/A σ -Siemens/ meter	$>0^a$

^arefer to section 3.5.2 for a complete description

Table 5. APM CSCI output data element requirements.

Name	Description	Type	Units	Source
i_{error}	Integer value that is returned if an error occurs in called routine	integer	N/A	APMINIT CSC XOINIT CSC
i_{xostp}	Index of output range step at which XO model is to be applied	integer	N/A	APMINIT CSC
j_{end}	Output height index at which valid propagation loss values end	integer	N/A	APMSTEP CSC
j_{start}	Output height index at which valid propagation loss values begin	integer	N/A	APMSTEP CSC
j_{xend}	Output height index at which valid XO propagation loss values end	integer	N/A	XOSTEP CSC
j_{xstart}	Output height index at which valid XO propagation loss values begin	integer	N/A	XOINIT CSC
$mpfl$	Propagation loss and factor array	integer	cB	APMSTEP CSC XOSTEP CSC
r_{out}	Current output range	real	meters	APMSTEP CSC XOSTEP CSC

3.3 CSCI INTERNAL INTERFACE REQUIREMENTS

Section 3.1 shows the relationship between the APM CSCI and its four CSCs: APMINIT, APMSTEP, XOINIT, and XOSTEP. The required internal interface between these four CSCs and the APM CSCI is left to the designer. However, Table 7 should be used as a guide to the required internal interfaces in the CSCI.

3.4 CSCI INTERNAL DATA REQUIREMENTS

The APM CSCI takes full advantage of Fortran 90 features, utilizing allocatable arrays for all internal and input arrays. This requires the NITES CSCI designer to correctly allocate and initialize all arrays necessary for input to the APM CSCI.

Due to the computational intensity of the APM CSCI, it may not be necessary or desirable to use the extreme capability of the APM CSCI for all applications. The variables n_{rout} and n_{zout} refer to the desired number of range and height output points for any one particular application, and will be specified when the APMINIT CSC is called.

One of the parameters returned to the NITES application from the APMINIT CSC is i_{error} . This is to allow for greater flexibility in how input data is handled within the NITES application.

Table 6 lists all possible errors that can be returned.

Table 6. APMINIT SU returned error definitions.

i_{error}	Definition
-6	Last range in terrain profile is less than r_{max} . Will only return this error if <i>lerr6</i> set to '.true.'.
-7	Specified cut-back angles (for user-defined height-finder antenna pattern) are not increasing.
-8	h_{max} is less than maximum height of terrain profile.
-9	Antenna height with respect to mean sea level is greater than maximum height h_{max} .
-10	Beamwidth is less than or equal to zero for directional antenna pattern.
-11	Number of antenna pattern or power reduction factors and angles is less than or equal to 1. For $i_{pat} = 6$, n_{facs} must be at least 1; for $i_{pat} = 7$, n_{facs} must be at least 2.
-12	Range of last environment profile given (for range-dependent case) is less than r_{max} . Will only return this error if <i>lerr12</i> set to '.true.'.
-13	Height of first level in any user-specified refractivity profile is greater than 0. First height must be at mean sea level (0.0) or < 0.0 if below mean sea level.
-14	Last gradient in any environment profile is negative.
-17	Range points of terrain profile are not increasing.
-18	First range value in terrain profile is not 0.
-19	Elevation angle specified is greater than 10° or less than -10° .
-25	Specified only the PE model to be used but did not specify maximum propagation angle th_{max} .
-41	Transmitter height is less than 1.5 meters.
-42	Minimum height input by user, h_{min} , is greater than maximum height, h_{max} .
-43	Transform size is greater than 2^{30} .
-44	Combination of frequency and antenna beamwidth results in antenna physically below the surface. Increase frequency or beamwidth for valid combination.
-45	Wind speed specified is greater and 10 m/s.

The logical variables *lerr6* and *lerr12*, when set to '.false.', allow the NITES application to bypass their associated errors as these are not critical to the operation of the APM CSCI.

The APM CSCI provides propagation loss and propagation factor for all heights and ranges when running in a full hybrid mode. When running in a partial hybrid mode, it provides propagation loss and factor for all heights, but not necessarily for all angles. Refer to Section 3.1 for environmental conditions under which each execution mode is automatically selected.

Absorption by atmospheric gases (oxygen and water vapor) may be important to some applications of the APM CSCI and is controlled by specifying a non-zero value for the absolute humidity, abs_{hum} , and the surface air temperature, t_{air} ; or likewise, specifying a non-zero value for the gaseous absorption attenuation rate, γ_a .

A particular application of the APM CSCI may or may not require the consideration of troposcatter effects within the propagation loss/factor calculations. For example, a radar evaluation most likely would not be influenced by troposcatter; while an ESM evaluation would. APM has the feature of including or not including the troposcatter calculation by setting a logical flag called T_{ropo} . Setting this flag to ‘.false.’ would omit the calculation. Setting this flag to ‘.true.’ would include the calculation. For the APM CSCI implementation within the NITES coverage and loss diagram applications, T_{ropo} must be set to ‘.true.’ so as to include the calculation.

APM also has the added capability to account for rough sea surface effects. Specifying a wind speed and a corresponding range will produce forward scatter results based on the Philips ocean-wave model for the rms wave height and the Miller-Brown reflection coefficient reduction factor. The capability also exists to allow variable wind speeds with range (Ref. 19).

APM by default will run in an “automatic” mode which, depending upon user-specified inputs, will choose the appropriate sub-models to use for a particular application. However, by setting the logical flag PE_{flag} to ‘.true.’ this will force APM to use only the PE sub-model for a particular NITES application. By default, this flag is set to ‘.false.’. If this flag is ‘.true.’ then the visible portion of the maximum PE propagation angle, th_{max} , (i.e., the maximum propagation angle the PE algorithm will accommodate in the field calculations) and the parameter, r_{mult} , must be specified. By default, r_{mult} is equal to 1; however th_{max} does not have a default value and must be explicitly defined. The parameter r_{mult} is a range step multiplier, allowing the user to vary the PE range step from the default calculated.

Use this option with caution as you must have some basic knowledge of PE algorithms and how they work to input proper combinations of maximum calculation angles and range steps for a given frequency. *When using this option, most error checking is bypassed and parameter limits can be over-ridden. Erroneous field values may result if a poorly chosen combination of th_{max} and r_{mult} are used.*

3.5 ADAPTATION REQUIREMENTS

3.5.1 Environmental Radio Refractivity Field Data Elements

The radio-refractivity field, i.e., the profiles of M-units versus height, must consist of vertical piece-wise linear profiles specified by couplets of height in meters with respect to mean sea level and modified refractivity (M-units) at multiple arbitrary ranges. All vertical profiles must contain the same number of vertical data points, and be specified such that each numbered data point corresponds to like-numbered points (i.e., features) in the other profiles. The first numbered data point of each profile must correspond to a height of zero mean sea level and the last numbered data point must correspond to a

height such that the modified refractivity for all greater heights is well represented by extrapolation using the two highest profile points specified.

With the inclusion of terrain and allowing the terrain profile to fall below mean sea level, refractivity profiles can also be provided in which the first level is less than 0 (or below mean sea level). For a terrain profile that falls below mean sea level at some point, the assumption is that the minimum height may be less than the first height in any refractivity profile specified. Therefore, an extrapolation flag, i_{extra} , must be specified to indicate how the APM CSCI should extrapolate from the first refractivity level to the minimum height along the terrain profile. Setting i_{extra} to 0 will cause the APM CSCI to extrapolate to the minimum height using a standard atmosphere gradient; setting i_{extra} to 1 will cause the APM CSCI to extrapolate to the minimum height using the gradient determined from the first two levels of the refractivity profile.

Within each profile, each numbered data point must correspond to a height greater than or equal to the height of the previous data point. Note that this requirement allows for a profile containing redundant data points. Note also that all significant features of the refractivity profiles must be specified, even if they are above the maximum output height specified for a particular application of APM.

The NITES CSCI application designer and the NITES operator share responsibility for determining appropriate environmental inputs. For example, a loss diagram may be used to consider a surface-to-surface radar detection problem. Since the operator is interested in surface-to-surface, he may truncate the profile assuming that effects from elevated ducting conditions are negligible. It may be, however, that the elevated duct does indeed produce a significant effect. The operator should ensure therefore, that the maximum height of the profile allows for the inclusion of all significant refractive features.

This specification allows a complicated refractivity field to be described with a minimum of data points. For example, a field in which a single trapping layer linearly descends with increasing range can be described with just two profiles containing only four data points each, frame (a) of Figure 7. In the same manner, other evolutions of refractive layers may be described. Frames (b) and (c) of Figure 7 show two possible scenarios for the development of a trapping layer. The scenario of choice is the one which is consistent with the true thermodynamical and hydrological layering of the atmosphere.

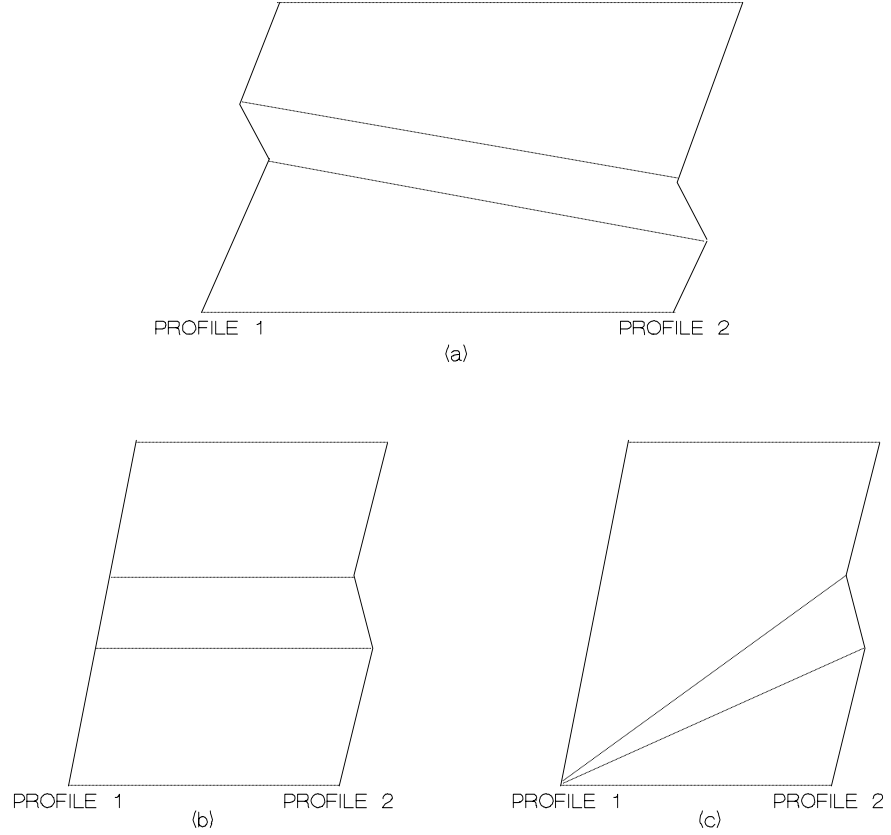


Figure 7. Idealized M-unit profiles (solid) and lines of interpolation (dashed).

Two external implementation data variables applicable to both the NITES operator and to the calling application designer are r_{max} , the maximum APM CSCI output range, and h_{max} , the maximum APM CSCI output height. These two parameters are required by the APM CSCI to determine the horizontal and vertical resolution, respectively, for internal range and height calculations based on the current values of n_{rout} and n_{zout} . Any value of r_{max} and h_{max} is allowed for the convenience of the NITES operator and the calling application designer, provided $r_{max} \geq 5$ km, and $h_{max} \geq 100$ m. For example, the NITES operator may desire a coverage diagram which extends to a range of 500 kilometers (km). In addition to accommodating the desires of the operator, specification of such a convenient maximum range eases the burden for the application designer in determining incremental tick marks for the horizontal axis of the display.

Provided the value of the parameter *lerr12* is set to ‘.false.’, if the furthest environment profile range is less than r_{max} , the APM CSCI will automatically create an environment profile at r_{max} equal to the last profile specified, making the environment homogeneous from the range of the last profile specified to r_{max} . For example, a profile is input with an accompanying range of 450 km. If the NITES operator chooses an r_{max} of 500 km, the APM CSCI will continue propagation loss/factor calculations to 500 km, keeping the refractivity environment homogeneous from 450 km to 500 km.

If *lerr12* is set to ‘.true.’ and the furthest environment profile range is less than r_{max} , then an error will be returned in i_{error} from the APMINIT CSC. This is to allow the NITES CSCI application designer greater flexibility in how environment data is handled.

3.5.2 Terrain Profile Data Element

The terrain profile must consist of linear piece-wise segments specified in terms of range/height pairs. All range values must be increasing, and the first terrain height value must be at range zero. General ground composition types can be specified (Table 4), along with corresponding ranges over which the ground type is to be applied. If ground type “User Defined” is specified ($igrnd_i = 7$), then numeric values of relative permittivity and conductivity must be given. If horizontal antenna polarization is specified, and if running a smooth surface case, the APM CSCI will assume perfect conductivity for the entire terrain profile and will ignore any information regarding ground composition. If vertical antenna polarization is specified, or if performing rough surface calculations, then information regarding ground composition must also be specified. If wind speed has been provided, then rough surface calculations will also be performed.

The maximum height, h_{max} , must always be greater than the minimum height, h_{min} by at least 100 meters. Also, a value of h_{max} must be given such that it is larger than the maximum elevation height along a specified terrain profile.

Provided *lerr6* is set to ‘.false.’, if the furthest range point in the terrain profile is less than r_{max} , the APM CSCI will automatically create a height/range pair as part of the terrain profile at r_{max} with elevation height equal to the last height specified in the profile, making the terrain profile flat from the range of the last profile point specified to r_{max} . For example, a terrain profile is input where the last height/range pair is 50 meters in height with an accompanying range of 95 km. If the NITES operator chooses an r_{max} of 100 km, the APM CSCI will continue propagation loss/factor calculations to 100 km, keeping the terrain profile flat from 95 km to 100 km with an elevation height of 50 m.

If *lerr6* is set to ‘.true.’ and the furthest range point is less than r_{max} , then an error is returned in i_{error} from the APMINIT SU. This is to allow the NITES CSCI application designer greater flexibility in how terrain data is handled.

3.6 SECURITY AND PRIVACY REQUIREMENTS

The security and privacy requirements are the same as those required by the target employing NITES CSCI.

3.7 CSCI ENVIRONMENTAL REQUIREMENTS

The APM CSCI must be able to operate in the same hardware and software environments that the target employing NITES CSCI operates.

3.8 COMPUTER RESOURCE REQUIREMENTS

Section 3.1.2.4 describes requirements for a Discrete Sine/Cosine Fast-Fourier Transform (DRST) SU. However, other sine FFT routines are available in the commercial market, and such a sine FFT may already be available within another NITES CSCI. The selection of which FFT is ultimately used by APM CSCI is left to the application designer as every sine FFT will have hardware and/or software performance impacts.

3.9 SOFTWARE QUALITY FACTORS

The primary required quality factors can be divided into the three categories - design, performance, and adaptation.

The quality factors for the design category should include correctness, maintainability, and verifiability. Correctness describes the extent to which the APM CSCI conforms to its requirements and is to be determined from the criteria – completeness, consistency, and/or traceability. Maintainability specifies the effort required to locate and fix an error in the APM CSCI. Maintainability is to be determined from the criteria – consistency, modularity, self-descriptiveness (self-documentation), and/or simplicity. Verifiability characterizes the effort required to test the APM CSCI to ensure that it performs its intended function. Verifiability is to be determined from the criteria – modularity, self-descriptiveness, and/or simplicity.

The quality factor for performance category is reliability, which depicts the confidence that can be placed in the APM CSCI calculations. Reliability is to be determined from the criteria – accuracy, anomaly management, auditability, consistency, and/or simplicity.

The quality factors for the adaptation category are portability and reusability. Portability determines how easy it is to transport the APM CSCI from one hardware

and/or software environment to another. Portability is to be determined from the criteria – application independence, modularity, and/or self-descriptiveness. Reusability illustrates how easy it is to convert the APM CSCI (or parts of the CSCI) for use in another application. Reusability is to be determined from the criteria – application independence, document accessibility, functional scope, generality, hardware independence, modularity, simplicity, self-descriptiveness, and/or system clarity.

Section A.A.1 defines the software quality criteria.

Only the software quality criteria completeness, consistency, and traceability can be analyzed. Their calculation is described in Section A.2. The other criteria have to be determined by either demonstration, test, or inspection.

3.10 DESIGN AND IMPLEMENTATION CONSTRAINTS

3.10.1 Implementation and Application Considerations

The calling NITES CSCI application will determine the employment of the APM CSCI. However, the intensive computational nature of the APM CSCI must be taken into consideration when designing an efficient calling application. For this reason, the APM CSCI should be designed with flexibility for various hardware suites and computer resource management considerations. As stated in Section 1, this APM CSCI applies only to a coverage and loss diagram application. The following highly recommended guidelines are provided to aid in the design of a coverage or loss diagram application which will most efficiently employ the APM CSCI.

The APM CSCI propagation loss calculations are independent of any target or receiver considerations; therefore, for any EM emitter, one execution of the APM CSCI may be used to create both a coverage diagram and a loss diagram. Since both execution time and computer memory allocation should be a consideration when employing this model, it is most efficient and appropriate to execute the APM CSCI for a particular EM system/environmental/terrain combination before executing any application. The output of the APM CSCI would be stored in a file which would be accessed by multiple applications.

For example, the NITES operator may desire a coverage diagram for one particular radar system. At the beginning of the coverage diagram application, a check would be made for the existence of a previously created APM CSCI output file appropriate for the EM system, environmental, and terrain conditions. If such a file exists, the propagation loss values would be read from the file and used to create the coverage diagram. If the file does not exist, the APM CSCI would be executed to create one. As the APM CSCI is executing, its output could be routed simultaneously to a graphics display device and a file. This file could then be used in the loss diagram application should the operator also choose it. Two distinct applications, therefore, are

achieved with only one execution of the APM CSCI. Additionally, should the operator desire an individual coverage diagram for each of multiple targets, or a single coverage diagram illustrating radar detection of a low-flying missile superimposed upon a coverage diagram illustrating his own radar's vulnerability as defined by the missile's ESM receiver, only a single execution of the APM CSCI would be required, thereby saving valuable computer resources.

3.10.2 Programming Language and Source Implementation

3.10.2.1 Programming Language

The ANSI Fortran 90 program language standard must be used in the development of the APM CSCI (Ref. 10). This standard consists of the specifications of the language Fortran. With certain limitations the syntax and semantics of the old International Standard commonly known as "FORTRAN 77" are contained entirely within this new International Standard. Therefore, any standard-conforming FORTRAN 77 program is standard conforming under the Fortran 90 Standard. Note that the name of this language, Fortran, differs from that in FORTRAN 77 in that only the first letter is capitalized. The Overview section of the International Standard describes the major additions to FORTRAN 77 in this International Standard. Section 1.3 of the International Standard specifies the bounds of the Fortran language by identifying both those items included and those items excluded. Section 1.4.1 describes the FORTRAN 77 compatibility of the International Standard with emphasis on four FORTRAN 77 features having different interpolations in the new International Standard. The International Standard provides facilities that encourage the design and the use of modular and reusable software.

Section 8.2 of the International Standard describes nine obsolescent features of FORTRAN 77 that are redundant and for which better methods are available in FORTRAN 77 itself. These nine obsolescent features should not be used. These obsolescent features are:

1. **Arithmetic IF** - use the **IF** statement.
2. Real and double precision **DO** control variables and **DO** loop control expressions – use integer.
3. Shared **DO** termination and termination on a statement other than **END DO** or **CONTINUE** – use an **END DO** or a **CONTINUE** statement for each **DO** statement.
4. Branching to an **END IF** statement from outside its IF block – branch to the statement following the **END IF**.
5. Alternate return.

6. **PAUSE** statement.
7. **ASSIGN** and assigned **GO TO** statements.
8. Assigned **FORMAT** specifiers.
9. cH (nH) edit descriptor.

Remedies for the last five obsolescent features are described in section 8.2 of the International standard.

3.10.2.2 Source Implementation

Ref. 14 by the Naval Oceanographic Office establishes a uniform standard for all software submitted by all contributors to them. It is recommended that the coding requirements set forth in Section 4 of that document be followed. Among these recommendations are:

1. Special non-ANSI features shall be avoided. Non-ANSI practices that are necessary must be documented in the code itself.
2. Maximum use should be made of existing commercially available FORTRAN callable libraries.
3. Programs shall be designed and coded using only five basic control structures – sequence of operations (assignment, add, ...), **IF THEN ELSE**, **DO WHILE**, **DO UNTIL**, and **CASE**.
4. Procedures or routines that make up a module shall not exceed an average of 100 executable statements per procedure or routine and shall not exceed a maximum of 200 executable statements in any procedure or routine.
5. Branching statements (**GO TOs**) shall only pass control to a statement that is in the same procedure or routine. Each **GO TO** must pass control only forward of its point of occurrence.
6. Naming conventions shall be uniform throughout the software. Program, subprogram, module, procedure, and data names shall be uniquely chosen to identify the applicable function performed. The naming convention for **COMMON** shall be consistent across the entire program.
7. Constants shall be defined not calculated (e.g., do not use $HALF = \frac{1}{2}$, use $HALF = 0.5$)

8. Mixed-mode numerical operations should be avoided whenever possible. When determined to be necessary, the use shall be explicit (*FLOAT*, *FIX*, or in assignment statement) and completely described in comments.
9. Each component of the software shall have a prologue containing the name of the program, subprogram, or function and any version number; purpose; inputs; outputs; list of routines that call this routine; complete list of routines called including intrinsic functions such as *ABS* and *FLOAT*; glossary; and method.
10. To facilitate program comprehension, comment statements shall be used throughout the program code.
11. The use of the **EQUIVALENCE** statement shall be restricted to those where it either improves the readability of the code or the efficiency of the program. If the **EQUIVALENCE** statement is used, it must be fully documented in the prologue and inline comment statements.
12. No machine-dependent techniques are allowed, unless there is no other way of performing the task.
13. Initialize every variable before use.
14. Do not depend on the values of “local” variables computed on a previous call to a routine.
15. Program structural indentation shall be used to improve readability and clarity.

3.11 PERSONNEL-RELATED REQUIREMENTS

Not applicable.

3.12 TRAINING-RELATED REQUIREMENTS

The employing target software personnel implementing this CSCI into the NITES CSCI will require training to become familiar with APM. This requirement should be met by this document and the companion Software Design Description (SDD) and Software Test Description (STD) documents.

3.13 OTHER REQUIREMENTS

None.

3.14 PRECEDENCE AND CRITICALITY OF REQUIREMENTS

The requirements presented in Sections 3.1 through 3.5 and Sections 3.8 through 3.10 have precedence over Sections 3.6, 3.7, 3.11, 3.12, and 3.13 and should be given equal weight.

4. QUALIFICATION PROVISIONS

N/A

5. REQUIREMENTS TRACEABILITY

5.1 SYSTEM TRACEABILITY

This section provides traceability of requirements between the APM CSCI and the NITES CSCI.

1. The APM CSCI environmental data requirements should be obtained from the Tactical Environmental Data System database (TEDS) within the NITES CSCI. The APM CSCI terrain data element requirements should be obtained from the Digital Terrain Elevation Database (DTED) within the NITES CSCI. The radar/communication system data element requirements should be obtained from the EM system database within the NITES CSCI.
2. The NITES CSCI requirement of propagation loss vs. range and height should be obtained from the APM CSCI.

5.2 DOCUMENTATION TRACEABILITY

This section provides the following types of traceability between the Software Requirements Specification (SRS), the Software Design Description (SDD), and the Software Test Description (STD):

1. Traceability between levels of requirements;
2. Traceability between the software requirements and software design;
3. Traceability between the software requirements and qualification test information obtained from the software testing.

This traceability of the Advanced Propagation Model is presented in two tables. The first table, Table 7 given here, presents the traceability between levels of SRS requirements. The second table (Table 6 in the Software Design Document) presents the traceability between the software requirements and software design.

Table 7. Requirements traceability matrix for the SRS.

Software Requirements Specification		Software Requirements Specification	
SRS Requirement Name	SRS Paragraph Number	SRS Requirement Name	SRS Paragraph Number
CSCI Capability Requirements	3.1	Advance Propagation Initialization (APMINIT) CSC	3.1.1
Advance Propagation Initialization (APMINIT) CSC	3.1.1	Allocate Arrays APM (ALLARRAY_APM) SU	3.1.1.1
Advance Propagation Initialization (APMINIT) CSC	3.1.1	Allocate Array RO (ALLARRAY_RO) SU	3.1.1.3
Advance Propagation Initialization (APMINIT) CSC	3.1.1	Allocate Array XORUF (XORUF) SU	3.1.1.4
Advance Propagation Initialization (APMINIT) CSC	3.1.1	Alpha Impedance Initialization (ALN_INIT) SU	3.1.1.5
Alpha Impedance Initialization (ALN_INIT) SU	3.1.1.5	Get Alpha Impedance (GETALN) SU	3.1.1.12
Get Alpha Impedance (GETALN) SU	3.1.1.12	Get Reflection Coefficient (GETREFCOEF) SU	3.1.2.10
Advance Propagation Initialization (APMINIT) CSC	3.1.1	Dielectric Initialization (DIEINIT) SU	3.1.1.7
Advance Propagation Initialization (APMINIT) CSC	3.1.1	FFT Parameters (FFTPAR) SU	3.1.1.8
Advance Propagation Initialization (APMINIT) CSC	3.1.1	Fill Height Arrays (FILLHT) SU	3.1.1.9
Fill Height Arrays (FILLHT) SU	3.1.1.9	Ray Trace to Output Range (TRACE_ROUT) SU	3.1.1.23
Advance Propagation Initialization (APMINIT) CSC	3.1.1	Gaseous Absorption (GASABS) SU	3.1.1.10
Advance Propagation Initialization (APMINIT) CSC	3.1.1	Get Effective Earth Radius Factor (GET_K) SU	3.1.1.11
Advance Propagation Initialization (APMINIT) CSC	3.1.1	Get Grazing Angle (GETGRAZE) SU	3.1.1.13
Get Grazing Angle (GETGRAZE) SU	3.1.1.13	DOSHIFT SU	3.1.2.3
Get Grazing Angle (GETGRAZE) SU	3.1.1.13	Free-Space Range Step (FRSTP) SU	3.1.2.7
Free-Space Range Step (FRSTP) SU	3.1.2.7	Fast-Fourier Transform (FFT) SU	3.1.2.6

Table 7. Requirements traceability matrix for the SRS. (Continued)

Software Requirements Specification		Software Requirements Specification	
SRS Requirement Name	SRS Paragraph Number	SRS Requirement Name	SRS Paragraph Number
Fast-Fourier Transform (FFT) SU	3.1.2.6	Discrete Sine/Cosine Transform (DRST) SU	3.1.2.4
Get Grazing Angle (GETGRAZE) SU	3.1.1.13	RD Trace (RDTRACE) SU	3.1.1.19
Get Grazing Angle (GETGRAZE) SU	3.1.1.13	Refractivity Interpolation (REFINTER) SU	3.1.2.14
Refractivity Interpolation (REFINTER) SU	3.1.2.14	Interpolate Profile (INTPROF) SU	3.1.1.16
Refractivity Interpolation (REFINTER) SU	3.1.2.14	Profile Reference (PROFREF) SU	3.1.1.18
Refractivity Interpolation (REFINTER) SU	3.1.2.14	Remove Duplicate Refractivity Levels (REMDUP) SU	3.1.1.21
Get Grazing Angle (GETGRAZE) SU	3.1.1.13	Spectral Estimation (SPECEST) SU	3.1.2.18
Spectral Estimation (SPECEST) SU	3.1.2.18	Discrete Sine/Cosine Transform (DRST) SU	3.1.2.4
Advance Propagation Initialization (APMINIT) CSC	3.1.1	Get Maximum Angle (GETTHMAX) SU	3.1.1.14
Get Maximum Angle (GETTHMAX) SU	3.1.1.14	FFT Parameters (FFTPAR) SU	3.1.1.8
Get Maximum Angle (GETTHMAX) SU	3.1.1.14	Ray Trace to Output Range (TRACE_ROUT) SU	3.1.1.23
Advance Propagation Initialization (APMINIT) CSC	3.1.1	Grazing Angle Interpolation (GRAZE_INT) SU	3.1.1.15
Advance Propagation Initialization (APMINIT) CSC	3.1.1	PE Initialization (PEINIT) SU	3.1.1.17
PE Initialization (PEINIT) SU	3.1.1.17	Allocate Array PE (ALLARRAY_PE) SU	3.1.1.2
PE Initialization (PEINIT) SU	3.1.1.17	Interpolate Profile (INTPROF) SU	3.1.1.16
PE Initialization (PEINIT) SU	3.1.1.17	Starter Field Initialization (XYINIT) SU	3.1.1.25
Starter Field Initialization (XYINIT) SU	3.1.1.25	Antenna Pattern (ANTPAT) SU	3.1.1.6
Starter Field Initialization (XYINIT) SU	3.1.1.25	Discrete Sine/Cosine Transform (DRST) SU	3.1.2.4

Table 7. Requirements traceability matrix for the SRS. (Continued)

Software Requirements Specification		Software Requirements Specification	
SRS Requirement Name	SRS Paragraph Number	SRS Requirement Name	SRS Paragraph Number
Advance Propagation Initialization (APMINIT) CSC	3.1.1	Profile Reference (PROFREF) SU	3.1.1.18
Advance Propagation Initialization (APMINIT) CSC	3.1.1	Refractivity Initialization (REFINIT) SU	3.1.1.20
Refractivity Initialization (REFINIT) SU	3.1.1.20	Profile Reference (PROFREF) SU	3.1.1.18
Refractivity Initialization (REFINIT) SU	3.1.1.20	Remove Duplicate Refractivity Levels (REMDUP) SU	3.1.1.21
Advance Propagation Initialization (APMINIT) CSC	3.1.1	Remove Duplicate Refractivity Levels (REMDUP) SU	3.1.1.21
Advance Propagation Initialization (APMINIT) CSC	3.1.1	Terrain Initialization (TERINIT) SU	3.1.1.22
Advance Propagation Initialization (APMINIT) CSC	3.1.1	Troposcatter Initialization (TROPOINT) SU	3.1.1.24
Troposcatter Initialization (TROPOINT) SU	3.1.1.24	Antenna Pattern(ANTPAT) SU	3.1.1.6
Troposcatter Initialization (TROPOINT) SU	3.1.1.24	Get Effective Earth Radius Factor (GET_K) SU	3.1.1.11
CSCI Capability Requirements	3.1	Advance Propagation Model Step (APMSTEP) CSC	3.1.2
Advance Propagation Model Step (APMSTEP) CSC	3.1.2	Airborne Hybrid Model(AIRBORNE) SU	3.1.2.1
Airborne Hybrid Model (AIRBORNE) SU	3.1.2.1	Antenna Pattern(ANTPAT) SU	3.1.1.6
Advance Propagation Model Step (APMSTEP) CSC	3.1.2	Flat-Earth Model (FEM) SU	3.1.2.5
Flat-Earth Model (FEM) SU	3.1.2.5	Antenna Pattern (ANTPAT) SU	3.1.1.6
Flat-Earth Model (FEM) SU	3.1.2.5	Get Reflection Coefficient (GETREFCOEF) SU	3.1.2.10
Advance Propagation Model Step (APMSTEP) CSC	3.1.2	Parabolic Equation Step (PESTEP) SU	3.1.2.12
Parabolic Equation Step (PESTEP) SU	3.1.2.12	Calculate Propagation Loss (CALCLOS) SU	3.1.2.2
Calculate Propagation Loss (CALCLOS) SU	3.1.2.2	Get Propagation Factor (GETPFAC) SU	3.1.2.9

Table 7. Requirements traceability matrix for the SRS. (Continued)

Software Requirements Specification		Software Requirements Specification	
SRS Requirement Name	SRS Paragraph Number	SRS Requirement Name	SRS Paragraph Number
Calculate Propagation Loss (CALCLOS) SU	3.1.2.2	Troposcatter (TROPOSCAT) SU	3.1.2.19
Troposcatter (TROPOSCAT) SU	3.1.2.19	Antenna Pattern (ANTPAT) SU	3.1.1.6
Parabolic Equation Step (PESTEP) SU	3.1.2.12	DOSHIFT SU	3.1.2.3
Parabolic Equation Step (PESTEP) SU	3.1.2.12	Free-Space Range Step (FRSTP) SU	3.1.2.7
Free-Space Range Step (FRSTP) SU	3.1.2.7	Fast-Fourier Transform (FFT) SU	3.1.2.6
Fast-Fourier Transform (FFT) SU	3.1.2.6	Discrete Sine/Cosine Transform (DRST) SU	3.1.2.4
Parabolic Equation Step (PESTEP) SU	3.1.2.12	FZLIM SU	3.1.2.8
FZLIM SU	3.1.2.8	Get Propagation Factor (GETPFAC) SU	3.1.2.9
FZLIM SU	3.1.2.8	Save Profile (SAVEPRO) SU	3.1.2.17
FZLIM SU	3.1.2.8	Spectral Estimation (SPECEST) SU	3.1.2.18
Spectral Estimation (SPECEST) SU	3.1.2.18	Discrete Sine/Cosine Transform (DRST) SU	3.1.2.4
Parabolic Equation Step (PESTEP) SU	3.1.2.12	Get Alpha Impedance (GETALN) SU	3.1.1.12
Get Alpha Impedance (GETALN) SU	3.1.1.12	Get Reflection Coefficient (GETREFCOEF) SU	3.1.2.10
Parabolic Equation Step (PESTEP) SU	3.1.2.12	Mixed Fourier Transform (MIXEDFT) SU	3.1.2.11
Mixed Fourier Transform (MIXEDFT) SU	3.1.2.11	Free-Space Range Step (FRSTP) SU	3.1.2.7
Free-Space Range Step (FRSTP) SU	3.1.2.7	Fast-Fourier Transform (FFT) SU	3.1.2.6
Fast-Fourier Transform (FFT) SU	3.1.2.6	Discrete Sine/Cosine Transform (DRST) SU	3.1.2.4

Table 7. Requirements traceability matrix for the SRS. (Continued)

Software Requirements Specification		Software Requirements Specification	
SRS Requirement Name	SRS Paragraph Number	SRS Requirement Name	SRS Paragraph Number
Parabolic Equation Step (PESTEP) SU	3.1.2.12	Refractivity Interpolation (REFINTER) SU	3.1.2.14
Refractivity Interpolation (REFINTER) SU	3.1.2.14	Interpolate Profile (INTPROF) SU	3.1.1.16
Refractivity Interpolation (REFINTER) SU	3.1.2.14	Profile Reference (PROFREF) SU	3.1.1.18
Refractivity Interpolation (REFINTER) SU	3.1.2.14	Remove Duplicate Refractivity Levels (REMDUP) SU	3.1.1.21
Advance Propagation Model Step (APMSTEP) CSC	3.1.2	Ray Optics Loss (ROLOSS) SU	3.1.2.16
Ray Optics Loss (ROLOSS) SU	3.1.2.16	Ray Optics Calculation (ROCALC) SU	3.1.2.15
Ray Optics Calculation (ROCALC) SU	3.1.2.15	Antenna Pattern (ANTPAT) SU	3.1.1.6
Ray Optics Calculation (ROCALC) SU	3.1.2.15	Get Reflection Coefficient (GETREFCOEF) SU	3.1.2.10
Ray Optics Calculation (ROCALC) SU	3.1.2.15	Ray Trace (RAYTRACE) SU	3.1.2.13
CSCI Capability Requirements	3.1	Extended Optics Initialization (XOINIT) CSC	3.1.3
Extended Optics Initialization (XOINIT) CSC	3.1.3	Advanced Propagation Model Clean (APMCLEAN) SU	3.1.3.1
Advanced Propagation Model Clean (APMCLEAN) SU	3.1.3.1	Discrete Sine/Cosine (DRST) SU	3.1.2.4
Extended Optics Initialization (XOINIT) CSC	3.1.3	Mean Filter (MEANFILT) SU	3.1.3.2
CSCI Capability Requirements	3.1	Extended Optics Step (XOSTEP) CSC	3.1.4
Extended Optics Step (XOSTEP) CSC	3.1.4	Advanced Propagation Model Clean (APMCLEAN) SU	3.1.3.1
Advanced Propagation Model Clean (APMCLEAN) SU	3.1.3.1	Discrete Sine/Cosine Transform (DRST) SU	3.1.2.4
Extended Optics Step (XOSTEP) CSC	3.1.4	Extended Optics (EXTO) SU	3.1.4.1

Table 7. Requirements traceability matrix for the SRS. (Continued)

Software Requirements Specification		Software Requirements Specification	
SRS Requirement Name	SRS Paragraph Number	SRS Requirement Name	SRS Paragraph Number
Extended Optics (EXTO) SU	3.1.4.1	Troposcatter (TROPOSCAT) SU	3.1.2.19
Troposcatter (TROPOSCAT) SU	3.1.2.19	Antenna Pattern (ANTPAT) SU	3.1.1.6
Extended Optics Step (XOSTEP) CSC	3.1.4	Flat-Earth Model (FEM) SU	3.1.2.5
Flat-Earth Model (FEM) SU	3.1.2.5	Antenna Pattern (ANTPAT) SU	3.1.1.6
Flat-Earth Model (FEM) SU	3.1.2.5	Get Reflection Coefficient (GETREFCOEF) SU	3.1.2.10
Extended Optics Step (XOSTEP) CSC	3.1.4	Ray Optics Loss (ROLOSS) SU	3.1.2.16
Ray Optics Loss (ROLOSS) SU	3.1.2.16	Ray Optics Calculation (ROCALC) SU	3.1.2.15
Ray Optics Calculation (ROCALC) SU	3.1.2.15	Antenna Pattern (ANTPAT) SU	3.1.1.6
Ray Optics Calculation (ROCALC) SU	3.1.2.15	Get Reflection Coefficient (GETREFCOEF) SU	3.1.2.10
Ray Optics Calculation (ROCALC) SU	3.1.2.15	Ray Trace (RAYTRACE) SU	3.1.2.13
CSCI Capability Requirements	3.1	CSCI External Interface Requirements	3.2
CSCI Capability Requirements	3.1	CSCI Internal Interface Requirements	3.3
CSCI Capability Requirements	3.1	CSCI Internal Data Requirements	3.4
CSCI Capability Requirements	3.1	Adaptation Requirements	3.5
CSCI Capability Requirements	3.1	Security and Privacy Requirements	3.6
CSCI Capability Requirements	3.1	CSCI Environmental Requirements	3.7
CSCI Capability Requirements	3.1	Computer Resource Requirements	3.8
CSCI Capability Requirements	3.1	Software Quality Factors	3.9
CSCI Capability Requirements	3.1	Design And Implementation Constraints	3.10
Design And Implementation Constraints	3.10	Implementation and Application Considerations	3.10.1

Table 7. Requirements traceability matrix for the SRS. (Continued)

Software Requirements Specification		Software Requirements Specification	
SRS Requirement Name	SRS Paragraph Number	SRS Requirement Name	SRS Paragraph Number
Design And Implementation Constraints	3.10	Programming Language And Source Code Implementation	3.10.2
Programming Language And Source Code Implementation	3.10.2	Programming Language	3.10.2.1
Programming Language And Source Code Implementation	3.10.2	Source Implementation	3.10.2.2
CSCI Capability Requirements	3.1	Personnel-Related Requirements	3.11
CSCI Capability Requirements	3.1	Training-Related Requirements	3.12
CSCI Capability Requirements	3.1	Other Requirements	3.13
CSCI Capability Requirements	3.1	Precedence and Criticality of Requirements	3.14

6. NOTES

Table 8 is a glossary of acronyms and abbreviations used within this document.
 Table 9 is a glossary of Fortran terms used within this document.

Table 8. Acronyms and abbreviations.

Term	Definition
ANSI	American National Standards Institute
APM	Advanced Propagation Model
cB	centibel
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
dB	decibel
EM	Electromagnetic
FFT	Fast-Fourier Transform
Fortran	Formula Translation
km	kilometers
m	meters
M	modified refractivity units
MHz	megahertz
N/A	not applicable
PE	Parabolic Equation
p-space	phase (angle) space
rad	radians
SDD	Software Design Description
SRS	Software Requirements Specification
STD	Software Test Description
SU	Software Unit
NITES	Naval Integrated Tactical Environmental Subsystem
z-space	height space

Table 9. Fortran terms.

Term	Action or Definitions
ABS	Absolute value function
Arithmetic IF	Transfers control to one of three statement labels, depending on the value of <i>expression</i>
ASSIGN	Assigns the value of a format or statement label to an integer variable
CASE	Marks the beginning of a block of statements executed if an item in a list of expressions matches the test expressions
COMMON	Allows two or more program units to directly share variables without having to pass them as arguments
CONTINUE	Does not have any effect
DO	Repeatedly executes the statements following the DO statement through the statement which marks the end of the loop
DO WHILE	Executes a block of statements repeatedly while a logical condition remains true
END DO	Terminates a DO or DO WHILE loop
END IF	Terminates a block of IF statements
EQUIVALENCE	Causes two or more variables or arrays to occupy the same memory location
FIX	Data type conversion function
FLOAT	Data type conversion function
FORMAT	Sets the format in which data is written to or read from a file
GO TO	Transfers execution to the statement label assigned to variable
IF	If expression is true, statement is executed; if expression is false, program execution continues with the next executable statement
IF THEN ELSE	If expression is true, statements in the IF block are executed; if expression is false, control is transferred to the next ELSE, ELSE IF, or END IF statement at the same IF level
PAUSE	Temporarily suspends program execution and allows you to execute operating system commands during the suspension

APPENDIX A

A.1 DEFINITIONS OF QUALITY FACTOR CRITERIA

The criteria for judging the quality factors of Section 3.9 have the following definitions:

1. Accuracy. The precision of computations and control;
2. Anomaly management. The degree to which the program detects failure in order to maintain consistency;
3. Application independence. The degree to which the program is independent of nonstandard programming language features, operating system characteristics, and other environmental constraints;
4. Auditability. The ease with which conformance to standards can be checked;
5. Completeness. The degree to which full implementation of required function has been achieved;
6. Consistency. The use of uniform design and documentation techniques throughout the software development project;
7. Document accessibility. The availability of documents describing the program components;
8. Functional scope. The generality of the feature set and capabilities of the program;
9. Generality. The breadth of potential application of program components;
10. Hardware independence. The degree to which the software is decoupled from the hardware on which it operates;
11. Modularity. The functional independence of program components;
12. Self- descriptiveness. The degree to which the source code provides meaningful documentation;
13. Simplicity. The degree to which a program can be understood without difficulty;

14. System clarity. The ease for which the feature set and capabilities of the system can be determined;
15. Traceability. The ability to trace a design representation or actual program component back to requirements.

A.2 SOFTWARE QUALITY METRICS

A.2.1 Completeness Criteria

The criteria completeness can be determined from the metric:

1. no ambiguous references (input, function, output);
2. all data references defined;
3. all referenced functions defined;
4. all defined functions used;
5. all conditions and processing defined for each decision point;
6. all defined and referenced calling sequences parameters agree;
7. all problem reports resolved;
8. design agrees with requirements;
9. code agrees with design;
10. (score 0 for any untrue statement; 1 otherwise); and
11. metric value = SUM (scores)/9.

A.2.2 Consistency Criteria

The criteria consistency can be determined from the metric : number of modules violating the design standard divided by the number of modules.

A.2.3 Traceability Criteria

The criteria traceability can be determined from the metric: number of itemized requirements traced divided by the total number of requirements.

SOFTWARE DESIGN DESCRIPTION
FOR THE
ADVANCED PROPAGATION MODEL CSCI
(Version 1.3.1)

9 August 2002

Prepared for:
Space and Naval Warfare Systems Command (PMW-155)
San Diego, CA

Prepared by:
Space and Naval Warfare Systems Center, San Diego
Atmospheric Propagation Branch (Code 2858)
49170 Propagation Path
San Diego, CA 92152-7385

CONTENTS

1. SCOPE	1
1.1 Identification	1
1.2 System Overview	1
1.3 Document Overview	1
2. REFERENCED DOCUMENTS	1
3. CSCI-WIDE DESIGN DECISIONS	3
4. CSCI ARCHITECTURE DESIGN	5
4.1 CSCI Components	5
4.2 Concept of Execution	9
4.3 Interface Design	11
4.3.1 Interface Identification and Diagrams	11
4.3.2 External Interface	11
4.3.3 Internal Interface	16
4.3.4 Internal Data	22
5. CSCI DETAILED DESIGN	24
5.1 Advance Propagation Model Initialization (APMINIT) CSC	24
5.1.1 Allocate Arrays APM (ALLARRAY_APM) SU	38
5.1.2 Allocate Array PE (ALLARRAY_PE) SU	41
5.1.3 Allocate Array RO (ALLARRAY_RO) SU	43
5.1.4 Allocate Array XORUF (ALLARRAY_XORUF) SU	44
5.1.5 Alpha Impedance Initialization (ALN_INIT) SU	46
5.1.6 Antenna Pattern (ANTPAT) SU	47
5.1.7 Dielectric Initialization (DIEINIT) SU	50
5.1.8 FFT Parameters (FFTPAR) SU	54
5.1.9 Fill Height Arrays (FILLHT) SU	56
5.1.10 Gaseous Absorption (GASABS) SU	60
5.1.11 Get Effective Earth Radius Factor (GET_K) SU	62
5.1.12 Get Alpha Impedance (GETALN) SU	66
5.1.13 Get Grazing Angle (GETGRAZE) SU	68
5.1.14 Get Maximum Angle (GETTHMAX) SU	71
5.1.15 Grazing Angle Interpolation (GRAZE_INT) SU	78
5.1.16 Interpolate Profile (INTPROF) SU	82
5.1.17 PE Initialization (PEINIT) SU	83
5.1.18 Profile Reference (PROFREF) SU	87
5.1.19 RD Trace (RDTRACE) SU	89
5.1.20 Refractivity Initialization (REFINIT) SU	94
5.1.21 Remove Duplicate Refractivity Levels (REMDUP) SU	100
5.1.22 Terrain Initialization (TERINIT) SU	101
5.1.23 Trace to Output Range (TRACE_ROUT) SU	105
5.1.24 Troposcatter Initialization (TROPOINT) SU	107
5.1.25 Starter Field Initialization (XYINIT) SU	109
5.2 Advanced Propagation Model Step (APMSTEP) CSC	111
5.2.1 Airborne Hybrid Model (AIRBORNE) SU	114
5.2.2 Calculate Propagation Loss (CALCLOS) SU	116
5.2.3 DOSHIFT SU	121
5.2.4 Discrete Sine/Cosine Fast-Fourier Transform (DRST) SU	122
5.2.5 Flat-earth Model (FEM) SU	123
5.2.6 Fast-Fourier Transform (FFT) SU	126
5.2.7 Free-Space Range Step (FRSTP) SU	126

5.2.8 FZLIM SU	127
5.2.9 Get Propagation Factor (GETPFAC) SU	129
5.2.10 Get Reflection Coefficient (GETREFCOEF) SU	130
5.2.11 Mixed Fourier Transform (MIXEDFT) SU	132
5.2.12 Parabolic Equation Step (PESTEP) SU	135
5.2.13 Ray Trace (RAYTRACE) SU	138
5.2.14 Refractivity Interpolation (REFINTER) SU	144
5.2.15 Ray Optics Calculation (ROCALC) SU	146
5.2.16 Ray Optics Loss (ROLOSS) SU	151
5.2.17 Save Profile (SAVEPRO) SU	155
5.2.18 Spectral Estimation (SPECEST) SU	156
5.2.19 Troposcatter (TROPOSCAT) SU	158
5.3 Extended Optics Initialization (XOINIT) CSC	162
5.3.1 APM Clean (APMCLEAN) SU	164
5.3.2 Mean Filter (MEANFILT) SU	167
5.4 Extended Optics Step (XOSTEP) CSC	168
5.4.1 Extended Optics (EXTO) SU	170
6. REQUIREMENTS TRACEABILITY	174
7. NOTES	178
7.1 APM CSCI Implementation and Application Considerations	178
7.2 Environmental Radio Refractivity Field Data Elements	178
7.3 Terrain Profile Data Element	181
7.4 Acronym and Abbreviations	181
7.5 SDD Variable Name, FORTRAN Variable Name Cross Reference	183
APPENDIX A: FORTRAN SOURCE CODE FOR APM CSCI	203

Figures

1. APM calculation regions	4
2. APM CSCI program flow	10
3. Idealized M-unit profiles (solid) and lines of interpolation (dashed)	180

Tables

Table 1. APM CSCI environmental data element requirements.	12
Table 2. APM CSCI external EM system data element requirements.	13
Table 3. APM CSCI external implementation constants.	14
Table 4. APM CSCI external terrain data element requirements.	15
Table 5. APM CSCI output data element requirements.	16
Table 6. APM internal interface design.	17
Table 7. APMINIT SU returned error definitions.	23
Table 8. APMINIT CSC input data element requirements.	34
Table 9. APMINIT CSC output data element requirements.	36
Table 10. ALLARRAY_APM SU input data element requirements.	40
Table 11. ALLARRAY_APM SU output data element requirements.	40
Table 12. ALLARRAY_PE SU input data element requirements.	42
Table 13. ALLARRAY_PE SU output data element requirements.	43
Table 14. ALLARRAY_RO input data element requirements.	44
Table 15. ALLARRAY_RO output data element requirements.	44

Table 16. ALLARRAY_XORUF SU input data element requirements.	45
Table 17. ALLARRAY_XORUF SU output data element requirements.	46
Table 18. ALN_INIT SU input data element requirements.	47
Table 19. ALN_INIT SU output data element requirements.	47
Table 20. ANTPAT SU input data element requirements.	50
Table 21. ANTPAT SU output data element requirements.	50
Table 22. DIEINIT SU input data element requirements.	54
Table 23. DIEINIT SU output data element requirements.	54
Table 24. FFTPAR SU input data element requirements.	56
Table 25. FFTPAR SU output data element requirements.	56
Table 26. FILLHT SU input data element requirements.	59
Table 27. FILLHT SU output data element requirements.	60
Table 28. GASABS SU input data element requirements.	62
Table 29. GASABS SU output data requirements.	62
Table 30. GET_K SU input data element requirements.	65
Table 31. GET_K SU output data element requirements.	65
Table 32. GETALN SU input data element requirements.	67
Table 33. GETALN SU output data element requirements.	68
Table 34. GETGRAZE SU input data element requirements.	70
Table 35. GETGRAZE SU input data element requirements.	71
Table 36. GETTHMAX SU input data element requirements.	77
Table 37. GETTHMAX SU output data element requirements.	78
Table 38. GRAZE_INT SU input data element requirements.	81
Table 39. GRAZE_INT SU output data element requirements.	82
Table 40. INTPROF SU input data element requirements.	82
Table 41. INTPROF SU output data element requirements.	82
Table 42. PEINIT SU input data element requirements.	86
Table 43. PEINIT SU output data element requirements.	86
Table 44. PROFREF SU input data element requirements.	89
Table 45. PROFREF SU output data element requirements.	89
Table 46. RDTRACE SU input data element requirements.	94
Table 47. RDTRACE SU output data elements requirements.	94
Table 48. REFINIT SU input data element requirements.	99
Table 49. REFINIT SU output data element requirements.	99
Table 50. REMDUP SU input data element requirements.	101
Table 51. REMDUP SU output data element requirements.	101
Table 52. TERINIT SU input data element requirements.	104
Table 53. TERINIT SU output data element requirements.	104
Table 54. TRACE_ROUT SU input data element requirements.	106
Table 55. TRACE_ROUT SU output data element requirements.	107
Table 56. TROPOINT SU input data element requirements.	109
Table 57. TROPOINT SU output data element requirements.	109
Table 58. XYINIT SU input data element requirements.	111
Table 59. XYINIT SU output data element requirements.	111
Table 60. APMSTEP CSC input data element requirements.	113
Table 61. APMSTEP CSC output data element requirements.	114
Table 62. AIRBORNE SU input data element requirements.	115
Table 63. AIRBORNE SU output data element requirements.	116

Table 64. CALCLOS SU input data element requirements.	119
Table 65. CALCLOS SU output data element requirements.	121
Table 66. DOSHIFT SU input data element requirements.	122
Table 67. DOSHIFT SU output data element requirements.	122
Table 68. DRST input data element requirements.	123
Table 69. DRST output data element requirements.	123
Table 70. FEM SU input data element requirements.	125
Table 71. FEM SU output data element requirements.	125
Table 72. FFT SU input data element requirements.	126
Table 73. FFT SU output data element requirements.	126
Table 74. FRSTP SU input data element requirements.	127
Table 75. FRSTP SU output data element requirements.	127
Table 76. FZLIM SU input data element requirements.	128
Table 77. FZLIM SU output data element requirements.	129
Table 78. GETPFAC SU input data element requirements.	130
Table 79. GETPFAC SU output data element requirements.	130
Table 80. GETREFCOEF SU input data element requirements.	131
Table 81. GETREFCOEF SU output data element requirements.	132
Table 82. MIXEDFT SU input data element requirements.	134
Table 83. MIXEDFT SU output data element requirements.	135
Table 84. PESTEP SU input data element requirements.	137
Table 85. PESTEP SU output data element requirements.	138
Table 86. RAYTRACE SU input data element requirements.	143
Table 87. RAYTRACE SU output data element requirements.	143
Table 88. REFINITER SU input data element requirements.	145
Table 89. REFINITER SU output data element requirements.	145
Table 90. RO region indices, angles, and ranges.	146
Table 91. ROCALC SU input data element requirements.	150
Table 92. ROCALC SU output data element requirements.	150
Table 93. ROLOSS SU input data element requirements.	154
Table 94. ROLOSS SU output data element requirements.	155
Table 95. ROLOSS SU save data element requirements.	155
Table 96. SAVEPRO SU input data element requirements.	156
Table 97. SAVEPRO output data element requirements.	156
Table 98. SPECEST SU input data element requirements.	157
Table 99. SPECEST output data element requirements.	158
Table 100. TROPOSCAT SU input data element requirements.	161
Table 101. TROPOSCAT SU output data element requirements.	162
Table 102. XOINIT CSC input data element requirements.	163
Table 103. XOINIT CSC output data element requirements.	164
Table 104. APMCLEAR CSC input data element requirements.	164
Table 105. APMCLEAR CSC output data element requirements.	167
Table 106. MEANFILT SU input data element requirements.	168
Table 107. MEANFILT SU output data element requirements.	168
Table 108. XOSTEP CSC input data element requirements.	169
Table 109. XOSTEP CSC output data element requirements.	169
Table 110. EXTO SU input data element requirements.	172
Table 111. EXTO SU output data element requirements.	174

Table 112. EXTO SU save data element requirements.	174
Table 113. Traceability matrix between the SRS and the SDD.	174
Table 114. Acronyms and abbreviations	182
Table 115. Variable name cross reference.	184

1. SCOPE

1.1 IDENTIFICATION

The Advanced Propagation Model (APM) Version 1.3.1 computer software configuration item (CSCI) calculates range-dependent electromagnetic (EM) system propagation loss within a heterogeneous atmospheric medium over variable terrain, where the radio-frequency index of refraction is allowed to vary both vertically and horizontally, also accounting for terrain effects along the path of propagation.

1.2 SYSTEM OVERVIEW

The APM CSCI model is designed to calculate propagation loss values as EM energy propagates through a laterally heterogeneous atmospheric medium where the index of refraction is allowed to vary both vertically and horizontally, also accounting for terrain effects along the path of propagation. Numerous Naval Integrated Tactical Environmental Subsystem (NITES) applications require EM-system propagation loss values. The APM model described by this document may be applied to two NITES applications, one which displays propagation loss on a range versus height scale (commonly referred to as a coverage diagram) and one which displays propagation loss on a propagation loss versus range/height scale (commonly referred to as a loss diagram).

1.3 DOCUMENT OVERVIEW

This document describes the design of the APM CSCI. An overview of the input software requirements is presented together with an overview of the CSCI design architecture and a detailed design description of each component of the CSCI.

2. REFERENCED DOCUMENTS

1. Bergland, G. D., "A Radix-eight Fast Fourier Transform Subroutine for Real-valued Series," *IEEE Trans. Audio and Electro-acoust.*, Vol. AU-17, pp. 138-144, 1969.
2. Cooley, J. W., P. A. W. Lewis and P. D. Welsh, "The Fast Fourier Transform Algorithm: Programming Considerations in the Calculation of Sine, Cosine and Laplace Transforms," *J. Sound Vib.*, Vol. 12, pp. 315-337, 1970.
3. Tappert, F. D., "The Parabolic Approximation Method," *Wave Propagation and Underwater Acoustics*, J. B. Keller and J. S. Papadakis, Eds., New York, Springer-Verlag, pp. 224-285, 1977.

4. International Radio Consulting Committee (CCIR) XVth Plenary Assembly Dubrovnik, 1986, "Propagation in Non-Ionized Media," *Recommendations and Reports of the CCIR, 1986*, Vol. V, International Telecommunications Union, Geneva, 1986.
5. Commander-In-Chief, Pacific Fleet Meteorological Requirement (PAC MET) 87-04, "Range Dependent Electromagnetic Propagation Models," 1987.
6. Dockery, G. D., "Modeling Electromagnetic Wave Propagation in the Troposphere Using the Parabolic Equation," *IEEE Trans. On Antennas and Propagat.*, Vol. 36, No. 10, pp. 1464-1470, October 1988.
7. Naval Oceanographic Office, "Software Documentation Standards and Coding Requirements for Environmental System Product Development," April 1990.
8. Kuttler, J. R. and G. D. Dockery., "Theoretical Description of the Parabolic Approximation/Fourier Split-Step Method of Representing Electromagnetic Propagation in the Troposphere," *Radio Sci.*, Vol. 26, pp. 381-393, 1991.
9. Dockery, G. D. and J. R. Kuttler. "An Improved Impedance-Boundary Algorithm for Fourier Split-Step Solutions of the Parabolic Wave Equation," *IEEE Trans. On Antennas and Propagat.*, Vol. 44, No. 12, pp. 1592-1599 December 1996
10. American National Standards Institute (ANSI), "Program Language Fortran Extended," 1992.
11. Patterson, W.L. and H. V. Hitney. "Radio Physical Optics CSCI Software Documents," Naval Command, Control and Ocean Surveillance Center, RDT&E Division, San Diego, CA, TD 2403, December 1992.
12. Barrios, A. E., "Terrain and Refractivity Effects on Non-Optical Paths," AGARD Conference Proceedings 543, Multiple Mechanism Propagation Paths (MMPPs): Their Characteristics and Influence on System Design, pp. 10-1 to 10-9, October 1993.
13. Barrios, A. E., "A Terrain Parabolic Equation Model for Propagation in the Troposphere," *IEEE Trans. Antennas Propagat.*, Vol. 42, pp. 90-98, January 1994.
14. Naval Oceanographic Office, "Software Documentation Standards for Environmental System Product Development," February 1996.

15. Barrios, A. E., "Terrain Parabolic Equation Model (TPEM) Version 1.5 User's Manual," Naval Command, Control and Ocean Surveillance Center, RDT&E Division, San Diego, CA, NRaD TD 2898, February 1996.
16. Sailors, D. B. and A. E. Barrios. "Terrain Parabolic Equation Model (TPEM) Computer Software Configuration Item (CSCI) Documents," Naval Command, Control and Ocean Surveillance Center, RDT&E Division, San Diego, CA, TD 2963, May 1997.
17. Sailors, D. B., A. E. Barrios, W. L. Patterson, and H. V. Hitney. "Advanced Propagation Model [Ver. 1.0] (APM) Computer Software Configuration Item (CSCI) Documents," SSC San Diego, San Diego, CA, TD 3033, August 1998.
18. Horst, M. M., F. B. Dyer, and M. T. Tuley. "Radar Sea Clutter Model," IEEE International Conference on Antennas and Propagation.
19. Miller, A. R., R. M. Brown, and E. Vegh, "New Derivation for the Rough-Surface Reflection Coefficient and for the Distribution of Sea-Wave Elevations" *Proc. IEEE*, Vol. 131, Part H, 2, pp 114-116, 1984.

3. CSCI-WIDE DESIGN DECISIONS

The required APM CSCI propagation model is a range-dependent true hybrid model that uses the complementary strengths of both Ray Optics (RO) and Parabolic Equation (PE) techniques to calculate propagation loss both in range and altitude.

The atmospheric volume is divided into regions which lend themselves to the application of the various propagation loss calculation methods. Figure 1 illustrates these regions.

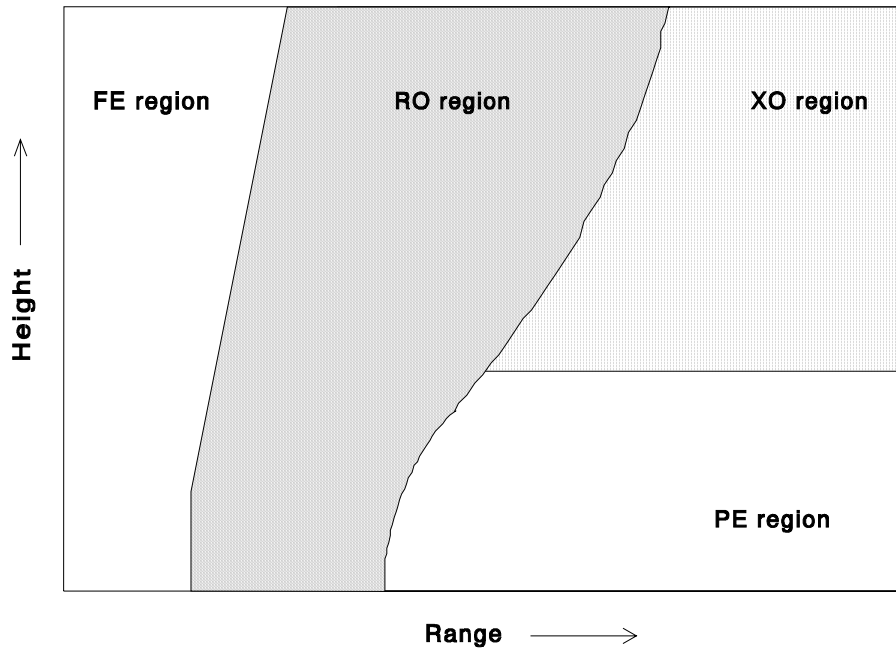


Figure 1. APM calculation regions.

For antenna elevation angles above 5 degrees or for ranges less than approximately 2.5 kilometers (km), a flat-earth (FE) ray-optics model is used. In this region, only receiver height is corrected for average refraction and earth curvature.

Within the RO region (as defined by a limiting ray), propagation loss is calculated from the mutual interference between the direct-path and surface-reflected ray components using the refractivity profile at zero range. Full account is given to focusing or de-focusing along both direct and reflected ray paths and to the integrated optical path length difference between the two ray paths, to give precise phase difference, and, hence, accurate coherent sums for the computation of propagation loss.

For the low-altitude region beyond the RO region, a PE approximation to the Helmholtz full wave equation is employed. The PE model allows for range-dependent refractivity profiles and variable terrain along the propagation path and uses a split step Fourier method for the solution of the PE. The PE model is run in the minimum region required to contain all terrain and trapping layer heights.

For the area beyond the RO region but above the PE region, an extended optics region (XO) is defined. Within the XO region, ray-optics methods which are initialized by the PE solution described below are used.

APM will run in three “execution” modes depending on environmental inputs. APM will use the FE, RO, XO, and PE models if the terrain profile is flat for the first 2.5 km and if the antenna height is less than or equal to 100 m. It will use only the XO and PE models if the terrain profile is *not* flat for the first 2.5 km and if the antenna height is less than or equal to 100 m. For applications in which the antenna height is greater than 100 meters, a combination of FE and PE methods are used. The FE model is used for all propagation angles greater than $\pm 5^\circ$ from the source and the PE model is used for angles less than $\pm 5^\circ$. By default, APM will automatically choose which mode of operation it will use for a specified set of inputs. However, the ability to run only the PE model for any case is allowed by setting a logical flag upon input.

The APM CSCI allows for horizontal and vertical antenna polarization, finite conductivity based on user-specified ground composition and dielectric parameters, and the complete range of EM system parameters and most antenna patterns required by NITES. APM also allows for gaseous absorption effects in all sub-models and computes troposcatter losses within the diffraction region and beyond.

The APM CSCI is divided into four main computer software components (CSC) and 47 additional software units (SUs). The purpose of the first CSC, the APMINIT CSC, is to interface with various SUs for the complete initialization of the APM CSCI. The purpose of the second CSC, the APMSTEP CSC, is to advance the entire APM CSCI algorithm one output range step, referencing various SUs to calculate the propagation loss at the current output range. The purpose of the XOINIT CSC is to initialize the range, height, and angle arrays in preparation for the XOSTEP CSC. The purpose of the fourth CSC, the XOSTEP CSC, is to advance the APM CSCI algorithm one output step from the top of the PE calculation region to the maximum output height specified, referencing various SUs to calculate the propagation output range.

4. CSCI ARCHITECTURE DESIGN

4.1 CSCI COMPONENTS

The APM CSCI is accessed by a subroutine call which provides, as global data elements, the values specified in Table 1 through Table 4. The source code for the APM CSCI is listed in Appendix A. The name and purpose for each CSC and SU are listed below.

The Advance Propagation Initialization (APMINIT) CSC interfaces with various SUs for the complete initialization of the APM CSCI. The APMINIT CSC component SUs include the following:

1. **Allocate Arrays APM (ALLARRAY_APM) SU.** Allocates and initializes all dynamically dimensioned arrays associated with APM terrain, refractivity, troposcatter, and general variable arrays.

2. **Allocate Array PE (ALLARRAY_PE) SU.** Allocates and initializes all dynamically dimensioned arrays associated with PE calculations.
3. **Allocate Array RO (ALLARRAY_RO) SU.** Allocates and initializes all dynamically dimensioned arrays associated with RO calculations.
4. **Allocate Array XO (ALLARRAY_XORUF) SU.** Allocates and initializes all dynamically dimensioned arrays associated with XO and rough surface calculations.
5. **Alpha Impedance Initialization (ALN_INIT) SU.** Initializes variables used in the Discrete Mixed Fourier Transform (DMFT) algorithm for finite conductivity and/or rough surface calculations.
6. **Antenna Pattern (ANTPAT) SU.** Calculates a normalized antenna gain (antenna pattern factor) for a specified antenna elevation angle.
7. **Dielectric Initialization (DIEINIT) SU.** Determines the conductivity and relative permittivity as a function of frequency in MHz based on general ground composition types.
8. **FFT Parameters (FFTPAR) SU.** Determines the required transform size based on the maximum PE propagation angle and the maximum height needed.
9. **Fill Height Arrays (FILLHT) SU.** Calculates the effective earth radius for an initial launch angle of 5° and fills an array with height values at each output range of the limiting sub-model, depending on which mode is being used.
10. **Gaseous Absorption (GASABS) SU.** Computes the specific attenuation based on air temperature and absolute humidity.
11. **Get Effective Earth Radius Factor (GET_K) SU.** Computes the effective earth radius factor and the effective earth radius.
12. **Get Alpha Impedance (GETALN) SU.** Computes the impedance term in the Leontovich boundary condition and the complex index of refraction for finite conductivity and vertical polarization calculations.
13. **Get Grazing Angle (GETGRAZE) SU.** Computes the grazing angles at each PE range step for subsequent use in rough sea surface calculations.
14. **Get Maximum Angle (GETTHMAX) SU.** Performs an iterative ray trace to determine the minimum angle required (based on the reflected ray) in obtaining a PE solution.

15. **Grazing Angle Interpolation (GRAZE_INT) SU.** Interpolates grazing angles at each PE range step based on angles computed from ray trace (takes precedence) and those computed from spectral estimation.
16. **Interpolate Profile (INTPROF) SU.** Performs a linear interpolation vertically with height on the refractivity profile.
17. **PE Initialization (PEINIT) SU.** Initializes all variables used in the PE model for subsequent calls to the PESTEP SU
18. **Profile Reference (PROFREF) SU.** Adjusts the current refractivity profile so that it is relative to a reference height.
19. **RD Trace (RDTRACE) SU.** Performs ray trace of many rays launched within an angle of $\pm 1.5^\circ$, storing grazing angles from these rays.
20. **Refractivity Initialization (REFINIT) SU.** Checks for valid environmental profile inputs and initializes refractivity arrays.
21. **Remove Duplicate Refractivity Levels (REMDUP) SU.** Removes any duplicate refractivity levels in the currently interpolated profile.
22. **Terrain Initialization (TERINIT) SU.** Examines and initializes terrain arrays for subsequent use in PE calculations.
23. **Trace to Output Range (TRACE_ROUT) SU.** Traces a single ray, whose launch angle is specified by the calling routine, to each output range.
24. **Troposcatter Initialization (TROPOINT) SU.** Initializes all variables and arrays needed for subsequent troposcatter calculations.
25. **Starter Field Initialization (XYINIT) SU.** Calculates the complex PE solution at range zero.

The Advanced Propagation Model Step (APMSTEP) CSC advances the entire APM CSCI algorithm one output range step, referencing various SUs to calculate the propagation loss at the current output range. The APMSTEP CSC component SUs include the following:

1. **Airborne Hybrid Model (AIRBORNE) SU.** Determines propagation loss based on flat-earth calculations for the direct ray path only for regions above and below the PE maximum propagation angle.

2. **Calculate Propagation Loss (CALCLOS) SU.** Determines propagation loss from the complex PE field at each output height point at the current output range.
3. **DOSHIFT SU.** Shifts the field by the number of bins, or PE mesh heights corresponding to the local ground height.
4. **Discrete Sine/Cosine Fast-Fourier Transform (DRST) SU.** Performs a sine or cosine transform, depending on the value of an integer flag provided by the calling SU, on both the real and imaginary components of the PE field which are passed separately.
5. **Flat-Earth Model (FEM) SU.** Computes propagation loss at a specified range based on flat-earth approximations.
6. **Fast-Fourier Transform (FFT) SU.** Separates the real and imaginary components of the complex PE field into two real arrays and then references the DRST SU.
7. **Free-Space Range Step (FRSTP) SU.** Propagates the complex PE solution field in free space by one range step.
8. **FZLIM SU.** Determines both the propagation factor (in dB) and the outgoing propagation angle at the top of the PE calculation region.
9. **Get Propagation Factor (GETPFAC) SU.** Determines the propagation factor at the specified height in dB.
10. **Get Reflection Coefficient (GETREFCOEF) SU.** Calculates the complex surface reflection coefficient, along with the magnitude and phase angle.
11. **Mixed Fourier Transform (MIXEDFT) SU.** Propagates the PE field in free space one PE range step, applying the Leontovich boundary condition, using the mixed Fourier transform as outlined by Kuttler and Dockery (Ref. 9).
12. **Parabolic Equation Step (PESTEP) SU.** Determines the next output range and begins an iterative loop to advance the PE solution such that for the current PE range, a PE solution is calculated from the solution at the previous PE range. This procedure is to be repeated until the output range is reached.
13. **Ray Trace (RAYTRACE) SU.** Traces a ray from a starting height and range with a specified starting elevation angle to a termination range.
14. **Refractivity Interpolation (REFINTER) SU.** Interpolates both horizontally and vertically on the modified refractivity profiles.

15. **Ray Optics Calculation (ROCALC SU).** Computes the RO components which will be needed in the calculation of propagation loss at a specified range and height within the RO region.
16. **Ray Optics Loss (ROLOSS) SU.** Calculates both the propagation loss and propagation factor values at a specified range and height based upon the components of magnitude for a direct-path and surface-reflected ray and the total phase lag angle between the two rays as determined by the ROCALC SU.
17. **Save Profile (SAVEPRO) SU.** Stores the refractivity profiles at each PE range step from the top of the PE region to the maximum user-specified height.
18. **Spectral Estimation (SPECEST) SU.** Determines, via spectral estimation, the outward propagation angle at the top of the PE calculation region.
19. **Troposcatter (TROPOSCAT) SU.** Determines the loss due to troposcatter and computes the appropriate loss between troposcatter and diffraction in the “transition” region using a method of “bold interpolation.”

The Extended Optics Initialization (XOINIT) CSC initializes the range, height, and angle arrays in preparation for the XOSTEP CSC. The XOINIT CSC component SUs include the following:

1. **Advanced Propagation Model Clean (APMCLEAN) SU.** Deallocates all dynamically dimensioned arrays used in one complete run of APM calculations.
2. **Mean Filter (MEANFILT) SU.** Performs an n-point average smoothing on any array passed to it.

The Extended Optics Step (XOSTEP) CSC advances the APM CSCI algorithm one output range step from the top of the PE calculation region to the maximum output height specified, referencing various SUs to calculate the propagation loss at the current output range. The XOSTEP CSC component SUs include the following:

1. **Extended Optics (EXTO) SU.** Calculates propagation loss and propagation factor, based on extended optics techniques, at the current output range.

4.2 CONCEPT OF EXECUTION

The program flow of the APM CSCI is illustrated in Figure 2. Note that the APM CSCI is shown within the context of a calling CSCI application such as one that generates a coverage or loss diagram. The efficient implementation of the APM CSCI will have far reaching consequences upon the design of an application CSCI beyond those mentioned in section 7.1. For example, Figure 2 shows checking for the existence of a previously created APM output file prior to the access of the APM CSCI. The application CSCI will have to consider if the atmospheric or terrain environment has changed since the APM output file was created or if any new height or range requirement is accommodated within the existing APM CSCI output file. Because these and many more considerations are beyond the scope of this document to describe, an application CSCI designer should work closely with the APM CSCI development agency in the implementation of the APM CSCI.

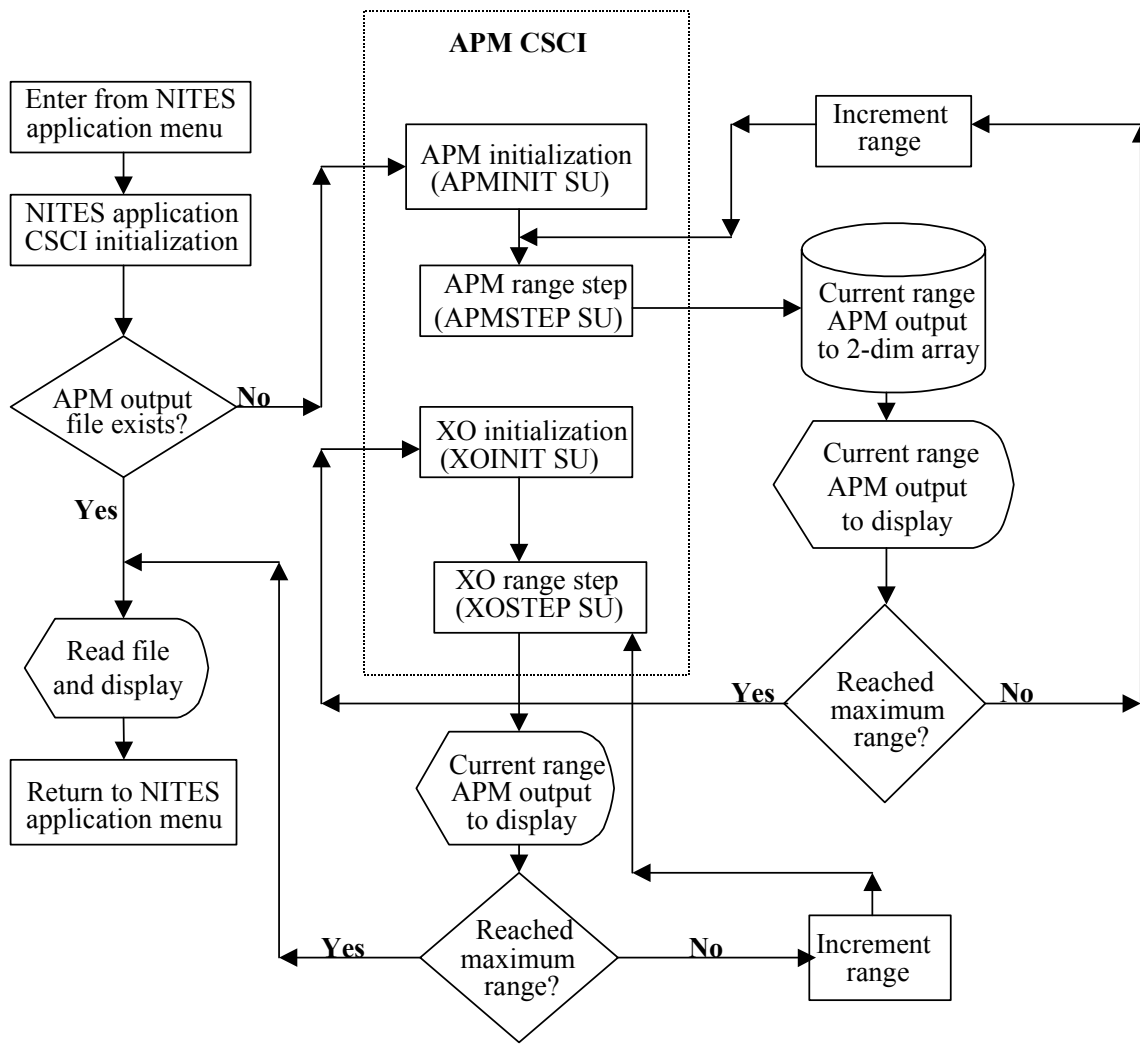


Figure 2. APM CSCI program flow.

4.3 INTERFACE DESIGN

4.3.1 Interface Identification and Diagrams

The APM CSCI interface design consists of one FORTRAN MODULE file for the external and internal data interface, FORTRAN CALL statements for both output data and internal interfacing, and several FORTRAN COMMON blocks for the internal interface. The MODULE file is called APM_MOD. This MODULE's statements provide several constants, COMMON blocks, and the dynamically allocated array names. The COMMON block names are: 1) APM_VAR, 2) ERRORFLAG, 3) INPUTVAR, 4) REFRACTIVITY, 5) SYSTEMVAR, 6) TERRAIN.

4.3.2 External Interface

The APM CSCI is accessed, through the APMINIT CSC, by a subroutine call from the NITES CSCI which should provide, as global data elements, the values specified in Table 1 through Table 4.

The APM CSCI external data elements, i.e., those data which must be provided by the calling NITES CSCI in the MODULE file prior to the APM CSCI execution can be divided into four classifications. The first is external data related to the atmospheric environment, specified within Table 1; the second is data related to the EM system, specified within Table 2; the third is data related to the implementation of the APM CSCI by the NITES CSCI, specified within Table 3; and the fourth is data related to the terrain information, specified within Table 4. Each table lists the type, units, and bounds of each data element. Table 5 specifies the output data of the APM CSCI model passed back to the calling CSCI via the FORTRAN CALL statements.

Table 1. APM CSCI environmental data element requirements.

Name	Description	Type	Units	Bounds
<i>refmsl</i>	Modified refractivity profile (dynamically allocated) array referenced to mean sea level	real	M	$\geq 0.0^a$
<i>hmsl</i>	Profile height (dynamically allocated) array	real	meters	See note b
<i>n_{prof}</i>	Number of refractivity profiles	integer	N/A	≥ 1
<i>lvlp</i>	Number of profile levels	integer	N/A	≥ 2
<i>rngprof</i>	Dynamically allocated array of ranges to each profile	real	meters	≥ 0.0
<i>abs_{hum}</i>	Surface absolute humidity	real	g/m ³	0 to 50 ^c
<i>t_{air}</i>	Surface air temperature	real	°C	-20 to 40 ^c
<i>γ_a</i>	Surface specific attenuation	real	dB/km	≥ 0.0
<i>i_{extra}</i>	Extrapolation flag for refractivity profiles entered in combination with terrain below mean sea level	integer	N/A	0 or 1
<i>n_w</i>	Number of wind speeds and corresponding ranges	integer	N/A	≥ 0.0
<i>rngwind</i>	Dynamically allocated array of ranges specified for each wind speed in <i>wind()</i> .	real	meters	≥ 0.0
<i>wind</i>	Dynamically allocated array of wind speeds.	real	meters/second	0.0 to 10.0

^aCouplets of height and modified refractivity associated with that height are referred to within this document as a refractivity profile.

^bAll heights in the refractivity profile must be steadily increasing.

^cThe CCIR gaseous absorption model implemented within APM provides a $\pm 15\%$ accuracy for absolute humidity and surface air temperature within these bounds. While values beyond these limits are allowed within APM, it should be noted this may result in less accurate attenuation rates calculated.

Table 2. APM CSCI external EM system data element requirements.

Name	Description	Type	Units	Bounds
μ_{bw}	Antenna vertical beam width	real	degree	.5 to 45
μ_o	Antenna elevation angle	real	degree	-10.0 to 10.0
f_{MHz}	EM system frequency	real	MHz	100.0 to 20,000.0
i_{pat}	Antenna pattern 1 = Omni-directional 2 = Gaussian 3 = Sine (X)/X 4 = Cosecant-squared 5 = Generic height-finder 6 = User-defined height-finder 7 = User-defined antenna pattern	integer	N/A	1 to 7
i_{pol}	Antenna polarization 0 = Horizontal 1 = Vertical	integer	N/A	0 to 1
ant_{ht}	Antenna height above local ground at range 0.0 m	real	meters	$\geq 1.5^a$
$hfang$	Dynamically allocated user-defined height-finder power reduction angle array ($i_{pat}=6$) or antenna pattern angles ($i_{pat}=7$)	real	degree	0.0 to 90.0 for $i_{pat}=6$ -90.0 to 90.0 for $i_{pat}=7$
$hffac$	Dynamically allocated user-defined power reduction factor array ($i_{pat}=6$) or antenna pattern factors ($i_{pat}=7$)	real	N/A	0.0 to 1.0
n_{fac}	Number of power reduction angles/factors for user-defined height-finder antenna pattern	integer	N/A	1 to 10

^aThe minimum antenna height will vary depending on the frequency and beamwidth according to the formula:

$$ant_{ht} \geq \text{maximum of} \left(1.5, 6 \frac{c_o}{f_{mhz} \mu_{bw}} \right)$$

where c_o is the speed of light x 10^{-6} m/s (299.79245).

Table 3. APM CSCI external implementation constants.

Name	Description	Type	Units	Bounds
h_{max}	Maximum height output for a particular application of APM	real	meters	$\geq 100.0^a$
h_{min}	Minimum height output for a particular application of APM	real	meters	$\geq 0.0^a$
$lerr6$	Logical flag to allow for error -6 to be bypassed	logical	N/A	'true.' or 'false.' ^b
$lerr12$	Logical flag to allow for error -12 to be bypassed	logical	N/A	'true.' or 'false.' ^b
n_{rout}	Number of range output points for a particular application of APM	integer	N/A	≥ 1
n_{zout}	Number of height output points for a particular application of APM	integer	N/A	≥ 1
PE_{flag}	Logical flag to enable only the PE model for a particular application of APM	logical	N/A	'true.' or 'false.' ^b
r_{max}	Maximum range output for a particular application of APM	real	meters	$\geq 5000.0^b$
r_{mult}	PE-range step multiplier	real	N/A	$> 0.0^b$
th_{max}	Visible portion of PE maximum calculation angle	real	degrees	$> 0.0^b$
T_{rope}	Logical flag to include troposcatter calculations.	integer	N/A	'true.' or 'false.'

^a refer to section 7.2 for a complete description.

^b refer to section 4.3.4 for a complete description.

Table 4. APM CSCI external terrain data element requirements.

Name	Description	Type	Units	Bounds
<i>terx</i>	Dynamically allocated terrain profile range array	real	meters	$\geq 0.0^a$
<i>tery</i>	Dynamically allocated terrain profile height array	real	meters	$\geq 0.0^a$
<i>i_{tp}</i>	Number of terrain profile points for a particular application of APM	integer	N/A	≥ 2
<i>i_{gr}</i>	Number of ground types for a particular application of APM	integer	N/A	$\geq 0.0^a$
<i>igrnd</i>	Array of ground composition types for a particular application of APM 0 = Sea water 1 = Fresh water 2 = Wet ground 3 = Medium dry ground 4 = Very dry ground 5 = Ice at -1° C 6 = Ice at -10° C 7 = User defined	integer	N/A	$0 \leq igrnd \leq 7^a$
<i>rgrnd</i>	Dynamically allocated array of ranges for which ground types are applied for a particular application of APM	real	meters	$\geq 0.0^a$
<i>dielec</i>	Dynamically allocated 2-dimensional array of relative permittivity (ϵ_r) and conductivity (σ) for a particular application of APM	real	ϵ_r - N/A σ -Siemens/ meter	$>0^a$

^arefer to section 7.3 for a complete description

Table 5. APM CSCI output data element requirements.

Name	Description	Type	Units	Source
i_{error}	Integer value that is returned if an error occurs in called routine	integer	N/A	APMINIT CSC XOINIT CSC
i_{xostp}	Index of output range step at which XO model is to be applied	integer	N/A	APMINIT CSC
j_{end}	Output height index at which valid propagation loss values end	integer	N/A	APMSTEP CSC
j_{start}	Output height index at which valid propagation loss values begin	integer	N/A	APMSTEP CSC
j_{xend}	Output height index at which valid XO propagation loss values end	integer	N/A	XOSTEP CSC
j_{xstart}	Output height index at which valid XO propagation loss values begin	integer	N/A	XOINIT CSC
$mpfl$	Propagation loss and factor array	integer	cB	APMSTEP CSC XOSTEP CSC
r_{out}	Current output range	real	meters	APMSTEP CSC XOSTEP CSC

4.3.3 Internal Interface

Section 4.2 shows the relationship between the APM CSCI and its four main CSCIs: APMINIT, AMPSTEP, XOINIT, and XOSTEP. This relationship is illustrated in Figure 2. The internal interface between these three CSCs and the APM CSCI is left to the design. However, the internal structure of the APM CSCI and its CSCs and SUs is shown in Table 6. The left two columns show the calling subroutines, and the right two columns the subroutines called. Columns 2 and 4 in Table 6 give the section number in Section 5 where more details about the various CSCs and SUs of the APM CSCI can be found.

Table 6. APM internal interface design.

Software Design Description		Software Design Description	
Software Design Description Name	SDD Paragraph Number	Software Design Description Name	SDD Paragraph Number
CSCI Detailed Design	5	Advance Propagation Initialization (APMINIT) CSC	5.1
Advance Propagation Initialization (APMINIT) CSC	5.1	Allocate Arrays APM (ALLARRAY_APM) SU	5.1.1
Advance Propagation Initialization (APMINIT) CSC	5.1	Allocate Array RO (ALLARRAY_RO) SU	5.1.3
Advance Propagation Initialization (APMINIT) CSC	5.1	Allocate Array XORUF (XORUF) SU	5.1.4
Advance Propagation Initialization (APMINIT) CSC	5.1	Alpha Impedance Initialization (ALN_INIT) SU	0
Alpha Impedance Initialization (ALN_INIT) SU	0	Get Alpha Impedance (GETALN) SU	5.1.12
Get Alpha Impedance (GETALN) SU	5.1.12	Get Reflection Coefficient (GETREFCOEF) SU	5.2.10
Advance Propagation Initialization (APMINIT) CSC	5.1	Dielectric Initialization (DIEINIT) SU	5.1.7
Advance Propagation Initialization (APMINIT) CSC	5.1	FFT Parameters (FFTPAR) SU	5.1.8
Advance Propagation Initialization (APMINIT) CSC	5.1	Fill Height Arrays (FILLHT) SU	5.1.9
Fill Height Arrays (FILLHT) SU	5.1.9	Ray Trace to Output Range (TRACE_ROUT) SU	5.1.23
Advance Propagation Initialization (APMINIT) CSC	5.1	Gaseous Absorption (GASABS) SU	5.1.10
Advance Propagation Initialization (APMINIT) CSC	5.1	Get Effective Earth Radius Factor (GET_K) SU	5.1.11
Advance Propagation Initialization (APMINIT) CSC	5.1	Get Grazing Angle (GETGRAZE) SU	5.1.13
Get Grazing Angle (GETGRAZE) SU	5.1.13	DOSHIFT SU	5.2.3
Get Grazing Angle (GETGRAZE) SU	5.1.13	Free-Space Range Step (FRSTP) SU	5.2.7
Free-Space Range Step (FRSTP) SU	5.2.7	Fast-Fourier Transform (FFT) SU	5.2.6

Table 6. APM internal interface design. (Continued)

Software Design Description		Software Design Description	
Software Design Description Name	SDD Paragraph Number	Software Design Description Name	SDD Paragraph Number
Fast-Fourier Transform (FFT) SU	5.2.6	Discrete Sine/Cosine Transform (DRST) SU	5.2.4
Get Grazing Angle (GETGRAZE) SU	5.1.13	RD Trace (RDTRACE) SU	5.1.19
Get Grazing Angle (GETGRAZE) SU	5.1.13	Refractivity Interpolation (REFINTER) SU	5.2.14
Refractivity Interpolation (REFINTER) SU	5.2.14	Interpolate Profile (INTPROF) SU	5.1.16
Refractivity Interpolation (REFINTER) SU	5.2.14	Profile Reference (PROFREF) SU	5.1.18
Refractivity Interpolation (REFINTER) SU	5.2.14	Remove Duplicate Refractivity Levels (REMDUP) SU	5.1.18
Get Grazing Angle (GETGRAZE) SU	5.1.13	Spectral Estimation (SPECEST) SU	5.2.18
Spectral Estimation (SPECEST) SU	5.2.18	Discrete Sine/Cosine Transform (DRST) SU	5.2.4
Advance Propagation Initialization (APMINIT) CSC	5.1	Get Maximum Angle (GETTHMAX) SU	5.1.14
Get Maximum Angle (GETTHMAX) SU	5.1.14	FFT Parameters (FFTPAR) SU	5.1.8
Get Maximum Angle (GETTHMAX) SU	5.1.14	Ray Trace to Output Range (TRACE_ROUT) SU	5.1.23
Advance Propagation Initialization (APMinit) CSC	5.1	Grazing Angle Interpolation (GRAZE_INT) SU	5.1.15
Advance Propagation Initialization (APMinit) CSC	5.1	PE Initialization (PEINIT) SU	5.1.17
PE Initialization (PEINIT) SU	5.1.17	Allocate Array PE (ALLARRAY_PE) SU	5.1.2
PE Initialization (PEINIT) SU	5.1.17	Interpolate Profile (INTPROF) SU	5.1.16
PE Initialization (PEINIT) SU	5.1.17	Starter Field Initialization (XYINIT) SU	5.1.25
Starter Field Initialization (XYINIT) SU	5.1.25	Antenna Pattern (ANTPAT) SU	5.1.6

Table 6. APM internal interface design. (Continued)

Software Design Description		Software Design Description	
Software Design Description Name	SDD Paragraph Number	Software Design Description Name	SDD Paragraph Number
Starter Field Initialization (XYINIT) SU	5.1.25	Discrete Sine/Cosine Transform (DRST) SU	5.2.4
Advance Propagation Initialization (APMINIT) CSC	5.1	Profile Reference (PROFREF) SU	5.1.18
Advance Propagation Initialization (APMINIT) CSC	5.1	Refractivity Initialization (REFINIT) SU	5.1.20
Refractivity Initialization (REFINIT) SU	5.1.20	Profile Reference (PROFREF) SU	5.1.18
Refractivity Initialization (REFINIT) SU	5.1.20	Remove Duplicate Refractivity Levels (REMDUP) SU	5.1.21
Advance Propagation Initialization (APMINIT) CSC	5.1	Remove Duplicate Refractivity Levels (REMDUP) SU	5.1.21
Advance Propagation Initialization (APMINIT) CSC	5.1	Terrain Initialization (TERINIT) SU	5.1.22
Advance Propagation Initialization (APMINIT) CSC	5.1	Troposcatter Initialization (TROPOINT) SU	5.1.24
Troposcatter Initialization (TROPOINT) SU	5.1.24	Antenna Pattern(ANTPAT) SU	5.1.6
Troposcatter Initialization (TROPOINT) SU	5.1.24	Get Effective Earth Radius Factor (GET_K) SU	5.1.11
CSCI Detailed Design	5	Advance Propagation Model Step (APMSTEP) CSC	5.2
Advance Propagation Model Step (APMSTEP) CSC	5.2	Airborne Hybrid Model (AIRBORNE) SU	5.2.1
Airborne Hybrid Model (AIRBORNE) SU	5.2.1	Antenna Pattern(ANTPAT) SU	5.1.6
Advance Propagation Model Step (APMSTEP) CSC	5.2	Flat-Earth Model (FEM) SU	5.2.5
Flat-Earth Model (FEM) SU	5.2.5	Antenna Pattern (ANTPAT) SU	5.1.6
Flat-Earth Model (FEM) SU	5.2.5	Get Reflection Coefficient (GETREFCOEF) SU	5.2.10
Advance Propagation Model Step (APMSTEP) CSC	5.2	Parabolic Equation Step (PESTEP) SU	5.2.12
Parabolic Equation Step (PESTEP) SU	5.2.12	Calculate Propagation Loss (CALCLOS) SU	5.2.2

Table 6. APM internal interface design. (Continued)

Software Design Description		Software Design Description	
Software Design Description Name	SDD Paragraph Number	Software Design Description Name	SDD Paragraph Number
Calculate Propagation Loss (CALCLOS) SU	5.2.2	Get Propagation Factor (GETPFAC) SU	5.2.9
Calculate Propagation Loss (CALCLOS) SU	5.2.2	Troposcatter (TROPOSCAT) SU	5.2.19
Troposcatter (TROPOSCAT) SU	5.2.19	Antenna Pattern (ANTPAT) SU	5.1.6
Parabolic Equation Step (PESTEP) SU	5.2.12	DOSHIFT SU	5.2.3
Parabolic Equation Step (PESTEP) SU	5.2.12	Free-Space Range Step (FRSTP) SU	5.2.7
Free-Space Range Step (FRSTP) SU	5.2.7	Fast-Fourier Transform (FFT) SU	5.2.6
Fast-Fourier Transform (FFT) SU	5.2.6	Discrete Sine/Cosine Transform (DRST) SU	5.2.4
Parabolic Equation Step (PESTEP) SU	5.2.12	FZLIM SU	5.2.8
FZLIM SU	5.2.8	Get Propagation Factor (GETPFAC) SU	5.2.9
FZLIM SU	5.2.8	Save Profile (SAVEPRO) SU	5.2.9
FZLIM SU	5.2.8	Spectral Estimation (SPECEST) SU	5.2.18
Spectral Estimation (SPECEST) SU	5.2.18	Discrete Sine/Cosine Transform (DRST) SU	5.2.4
Parabolic Equation Step (PESTEP) SU	5.2.12	Get Alpha Impedance (GETALN) SU	5.1.12
Get Alpha Impedance (GETALN) SU	5.1.12	Get Reflection Coefficient (GETREFCOEF) SU	5.2.10
Parabolic Equation Step (PESTEP) SU	5.2.12	Mixed Fourier Transform (MIXEDFT) SU	5.2.11
Mixed Fourier Transform (MIXEDFT) SU	5.2.11	Free-Space Range Step (FRSTP) SU	5.2.7
Free-Space Range Step (FRSTP) SU	5.2.7	Fast-Fourier Transform (FFT) SU	5.2.6
Fast-Fourier Transform (FFT) SU	5.2.6	Discrete Sine/Cosine Transform (DRST) SU	5.2.4

Table 6. APM internal interface design. (Continued)

Software Design Description		Software Design Description	
Software Design Description Name	SDD Paragraph Number	Software Design Description Name	SDD Paragraph Number
Parabolic Equation Step (PESTEP) SU	5.2.12	Refractivity Interpolation (REFINTER) SU	5.2.14
Refractivity Interpolation (REFINTER) SU	5.2.14	Interpolate Profile (INTPROF) SU	5.1.16
Refractivity Interpolation (REFINTER) SU	5.2.14	Profile Reference (PROFREF) SU	5.1.18
Refractivity Interpolation (REFINTER) SU	5.2.14	Remove Duplicate Refractivity Levels (REMDUP) SU	5.1.21
Advance Propagation Model Step (APMSTEP) CSC	5.2	Ray Optics Loss (ROLOSS) SU	5.2.16
Ray Optics Loss (ROLOSS) SU	5.2.16	Ray Optics Calculation (ROCALC) SU	5.2.15
Ray Optics Calculation (ROCALC) SU	5.2.15	Antenna Pattern (ANTPAT) SU	5.1.6
Ray Optics Calculation (ROCALC) SU	5.2.15	Get Reflection Coefficient (GETREFCOEF) SU	5.2.10
Ray Optics Calculation (ROCALC) SU	5.2.15	Ray Trace (RAYTRACE) SU	5.2.13
CSCI Detailed Design	5	Extended Optics Initialization (XOINIT) CSC	5.3
Extended Optics Initialization (XOINIT) CSC	5.3	Advanced Propagation Model Clean (APMCLEAN) SU	5.3.1
Advanced Propagation Model Clean (APMCLEAN) SU	5.3.1	Discrete Sine/Cosine (DRST) SU	5.2.4
Extended Optics Initialization (XOINIT) CSC	5.3	Mean Filter (MEANFILT) SU	5.3.2
CSCI Detailed Design	5	Extended Optics Step (XOSTEP) CSC	5.4
Extended Optics Step (XOSTEP) CSC	5.4	Advanced Propagation Model Clean (APMCLEAN) SU	5.3.1
Advanced Propagation Model Clean (APMCLEAN) SU	5.3.1	Discrete Sine/Cosine Transform (DRST) SU	5.2.4
Extended Optics Step (XOSTEP) CSC	5.4	Extended Optics (EXTO) SU	5.4.1

Table 6. APM internal interface design. (Continued)

Software Design Description		Software Design Description	
Software Design Description Name	SDD Paragraph Number	Software Design Description Name	SDD Paragraph Number
Extended Optics (EXTO) SU	5.4.1	Troposcatter (TROPOSCAT) SU	5.2.19
Troposcatter (TROPOSCAT) SU	5.2.19	Antenna Pattern (ANTPAT) SU	5.1.6
Extended Optics Step (XOSTEP) CSC	5.4	Flat-Earth Model (FEM) SU	5.2.5
Flat-Earth Model (FEM) SU	5.2.5	Antenna Pattern (ANTPAT) SU	5.1.6
Flat-Earth Model (FEM) SU	5.2.5	Get Reflection Coefficient (GETREFCOEF) SU	5.2.10
Extended Optics Step (XOSTEP) CSC	5.4	Ray Optics Loss (ROLOSS) SU	5.2.16
Ray Optics Loss (ROLOSS) SU	5.2.16	Ray Optics Calculation (ROCALC) SU	5.2.15
Ray Optics Calculation (ROCALC) SU	5.2.15	Antenna Pattern (ANTPAT) SU	5.1.6
Ray Optics Calculation (ROCALC) SU	5.2.15	Get Reflection Coefficient (GETREFCOEF) SU	5.2.10
Ray Optics Calculation (ROCALC) SU	5.2.15	Ray Trace (RAYTRACE) SU	5.2.13

4.3.4 Internal Data

The APM CSCI takes full advantage of Fortran 90 features, utilizing allocatable arrays for all internal and input arrays. This requires the NITES CSCI designer to correctly allocate and initialize all arrays necessary for input to the APM CSCI.

Due to the computational intensity of the APM CSCI, it may not be necessary or desirable to use the extreme capability of the APM CSCI for all applications. The variables n_{rout} and n_{zout} refer to the desired number of range and height output points for any one particular application, and will be specified when the APMINIT CSC is called.

One of the parameters returned to the NITES application from the APMINIT CSC is i_{error} . This allows for greater flexibility in how input data is handled within the NITES application. Table 7 lists all possible errors that can be returned.

The logical variables *lerr6* and *lerr12*, when set to ‘.false.’, allow the NITES application to bypass their associated errors as these are not critical to the operation of the APM CSCI.

Table 7. APMINIT SU returned error definitions.

<i>i_{error}</i>	Definition
-6	Last range in terrain profile is less than r_{max} . Will only return this error if <i>lerr6</i> set to ‘.true.’.
-7	Specified cut-back angles (for user-defined height-finder antenna pattern) are not increasing.
-8	h_{max} is less than maximum height of terrain profile.
-9	Antenna height with respect to mean sea level is greater than maximum height h_{max} .
-10	Beamwidth is less than or equal to zero for directional antenna pattern.
-11	Number of antenna pattern or power reduction factors and angles is less than or equal to 1. For $i_{pat} = 6$, n_{fac} must be at least 1; for $i_{pat} = 7$, n_{fac} must be at least 2.
-12	Range of last environment profile given (for range-dependent case) is less than r_{max} . Will only return this error if <i>lerr12</i> set to ‘.true.’.
-13	Height of first level in any user-specified refractivity profile is greater than 0. First height must be at mean sea level (0.0) or < 0.0 if below mean sea level.
-14	Last gradient in any environment profile is negative.
-17	Range points of terrain profile are not increasing.
-18	First range value in terrain profile is not 0.
-19	Elevation angle specified is greater than 10° or less than -10°.
-25	Specified only the PE model to be used but did not specify maximum propagation angle th_{max} .
-41	Transmitter height is less than 1.5 meters.
-42	Minimum height input by user, h_{min} , is greater than maximum height, h_{max} .
-43	Transform size is greater than 2^{30} .
-44	Combination of frequency and antenna beamwidth results in antenna physically below the surface. Increase frequency or beamwidth for valid combination.
-45	Wind speed specified is greater and 10 m/s.

The APM CSCI provides propagation loss and factor for all heights and ranges when running in a full hybrid mode. When running in a partial hybrid mode, it provides propagation loss and factor for all heights, but not necessarily for all angles. Finally, it will be limited in both height and angle coverage when running in a PE-only mode. Refer to Section 7.1 for environmental conditions under which each execution mode is automatically selected.

Absorption by atmospheric gases (oxygen and water vapor) may be important to some applications of the APM CSCI and is controlled by specifying a non-zero value for the absolute humidity, abs_{hum} , and the surface air temperature, t_{air} ; or likewise, specifying a non-zero value for the gaseous absorption attenuation rate, γ_a .

A particular application of the APM CSCI may or may not require the consideration of troposcatter effects within the propagation loss calculations. For example, a radar evaluation most likely would not be influenced by troposcatter; while an ESM evaluation would. APM has the feature of including or not including the troposcatter calculation by setting a logical flag called T_{ropo} . Setting this flag to ‘.false.’ would omit the calculation. Setting this flag to ‘.true.’ would include the calculation. For the APM CSCI implementation within the NITES coverage and loss diagram applications, T_{ropo} must be set equal to ‘.true.’ so as to include the calculation.

APM by default will run in an “automatic” mode which, depending upon user-specified inputs, will choose the appropriate sub-models to use for a particular application. However, by setting the logical flag PE_{flag} to ‘.true.’ this will force APM to use only the PE sub-model for a particular NITES application. By default, this flag is set to ‘.false.’. If this flag is ‘.true.’, then the visible portion of the maximum PE propagation angle, th_{max} , (i.e., the maximum propagation angle the PE algorithm will accommodate in the field calculations) and the parameter, r_{mult} , must be specified. By default, r_{mult} is equal to 1; however, th_{max} does not have a default value and must be explicitly defined. The parameter r_{mult} is a range step multiplier, allowing the user to vary the PE range step from the default calculated.

Use this option with caution as you must have some basic knowledge of PE algorithms and how they work to input proper combinations of maximum calculation angles and range steps for a given frequency. *When using this option, most error checking is bypassed and parameter limits can be over-ridden. Erroneous field values may result if a poorly chosen combination of th_{max} and r_{mult} are used.*

5. CSCI DETAILED DESIGN

A description of each component of the APM CSCI is provided in the following subsections.

5.1 ADVANCE PROPAGATION MODEL INITIALIZATION (APMINIT) CSC

The purpose of the APMINIT CSC is to interface with various SUs for the complete initialization of the APM CSCI.

Upon entering the APMINIT CSC, several variables are initialized. All internal logical flags controlling certain environmental calculations and errors are initialized.

The wind speeds and ranges are checked to see if they fall within the specified bounds. If not, then an error is returned.

Next, the absorption calculation flag, k_{abs} , is set to 1 if the air temperature, t_{air} , or the absolute humidity, abs_{hum} , are non-zero. If an attenuation rate is specified ($\gamma_a \neq 0$), then k_{abs} is set to 2. If the user specifies both t_{air} and abs_{hum} to be 0, then k_{abs} is set equal to 0, in which case no absorption losses are computed.

Next, if running the APM CSCI under the hybrid mode ($PE_{flag} = \text{'false.'}$), then the antenna height, the maximum output range, r_{max} , the maximum output height, h_{max} , and the minimum output height, h_{min} , are checked for valid numerical values. If the antenna height is below 1.5 meters, then i_{error} is set to -41 and the APMINIT CSC is exited. r_{max} is set to the value specified from the calling CSCI or 5000 meters, whichever is greater; and h_{max} is set to the value specified from the calling CSCI or 100 meters, whichever is greater. If h_{min} is greater than h_{max} , then i_{error} is set to -42 and the APMINIT CSC is exited. If the maximum output range and minimum and maximum output height values are valid, then the APMINIT CSC proceeds to the next step.

The atmospheric volume must be “covered” or resolved with a mesh of calculation points that will, as a matter of routine, exceed the height/range resolution requirements of the particular application of the APM CSCI. The height and range mesh size per APM CSCI output point, Δz_{out} and Δr_{out} , respectively, are calculated from the number of APM output points and the maximum range and height as follows:

$$\Delta r_{out} = \frac{r_{max}}{n_{rout}},$$

$$\Delta z_{out} = \frac{h_{max} - h_{min}}{n_{zout}}.$$

The number of terrain range/height pairs, i_{tpa} , used for internal calculations is initialized to 1 plus the user-specified number of range/height pairs, i_{tp} . The ALLARRAY_APM SU is then referenced to dynamically allocate and initialize all arrays associated with terrain, refractivity, troposcatter, and general variable arrays. If an error has occurred while allocating memory, i_{error} is returned with a non-zero value and the CSC is exited; otherwise, the CSC proceeds to the next step.

Next, the constants used to determine the antenna pattern factor are computed. First, if a user-defined height-finder antenna pattern has been specified ($i_{pat} = 6$), along with power cut-back angles and factors, then the angles are converted to radians and stored in array $hfangr$. If the cut-back angles are not steadily increasing, i_{error} is set to -7 and the CSC is exited; otherwise, the CSC proceeds with the next step.

If a directional antenna pattern has been specified, the antenna vertical beamwidth in degrees, μ_{bw} , is checked for extremely small beamwidth values. If the value is less than or equal to 10^{-4} , i_{error} is set to -10 and the CSC is exited. If the elevation angle is greater than 10° or less than -10° , then i_{error} is set to -19 and the CSC is exited; otherwise, the CSC proceeds with the next step.

The antenna beamwidth and elevation angles are converted to radians (μ_{bwr} and μ_{or} , respectively) and the following variables, ant_{fac} and μ_{max} , for use in the ANTPAT SU are determined as follows. If the antenna pattern is Gaussian ($i_{pat}=2$), then ant_{fac} is given by

$$ant_{fac} = \frac{.34657359}{\left[\text{SIN} \left(\frac{\mu_{bwr}}{2} \right) \right]^2}.$$

If the antenna pattern is Sin(X)/X ($i_{pat} = 3$), or a generic height finder ($i_{pat} = 5$), then ant_{fac} is given by

$$ant_{fac} = \frac{1.39157}{\text{SIN} \left(\frac{\mu_{bwr}}{2} \right)},$$

and μ_{max} is given by

$$\mu_{max} = \text{TAN}^{-1} \left(\frac{\pi}{ant_{fac} \sqrt{1 - \left(\frac{\pi}{ant_{fac}} \right)^2}} \right).$$

If running the APM CSCI in hybrid mode, the antenna height is next checked for a valid height based on the combination of frequency and beamwidth specified. The antenna height must not be less than the radius of the antenna and is bounded by

$$ant_{ht} \geq \text{maximum of} \left(1.5, .6 \frac{c_o}{f_{mhz} \mu_{bw}} \right),$$

where c_o is the speed of light (299.79×10^6 m/s).

Next, the TERINIT SU is referenced to initialize all terrain profile and associated arrays. If an error has occurred while in the TERINIT SU, i_{error} is returned with a non-zero value and the CSC is exited; otherwise, the CSC proceeds with the next step.

If vertical polarization and/or rough surface calculations are required (i.e., a non-zero wind speed is specified), then the flag, i_{alg} , indicating which DMFT algorithm to use, is set to 1 (central difference) for frequencies less than 400 MHz, and is set to 2 (backward difference) for frequencies greater than 400 MHz. The default value for i_{alg} is 0, in which case, no DMFT algorithm is used for the particular APM application.

Arrays containing all output ranges, $rngout$, 20 times the logarithm of all output ranges, $rlogo$, and the square of the output ranges, $rsqrd$, are initialized accordingly:

$$\begin{aligned} rngout_i &= i \Delta r_{out} , \\ rlogo_i &= 20 \text{ LOG}_{10} (i \Delta r_{out}) , \quad i = 1, 2, \dots, n_{rout} \\ rsqrd_i &= (i \Delta r_{out})^2 . \end{aligned}$$

The minimum power of 2 transform, ln_{min} , is next initialized to 10. If the PE-only option is not activated ($PE_{flag} = \text{'false.'}$), then the execution mode in which the APM CSCI will operate is determined. Based on inputs, it determines whether to use the airborne hybrid mode ($i_{hybrid}=0$), full hybrid mode ($i_{hybrid}=1$), or partial hybrid mode ($i_{hybrid}=2$). For antenna heights greater than 100 meters above the local ground height, i_{hybrid} is set equal to 0. For antenna heights less than 100 meters, i_{hybrid} is initialized to 1. If performing a terrain case ($f_{ter} = \text{'true.'}$) and the first 2500 meters of the terrain profile is not flat, then i_{hybrid} is set equal to 2. If running in full hybrid mode ($i_{hybrid}=1$), then the ALLARRAY_RO SU is referenced to allocate and initialize all arrays associated with RO calculations.

The variable y_{ref} is initialized to 0. If a terrain profile has been specified ($f_{ter} = \text{'true.'}$), then y_{ref} is set equal to ty_1 . Next, the output height arrays $zout$ and zro are initialized as follows:

$$\begin{aligned} zout_i &= hm_{ref} + i \Delta z_{out} , \quad i = 1, 2, \dots, n_{zout} . \\ zro_i &= zout_i - y_{ref} \end{aligned}$$

Next, the REFINIT SU is referenced to initialize all refractivity associated arrays. If an error has occurred while in the REFINIT SU, i_{error} is returned with a non-zero value and the CSC is exited; otherwise, the CSC proceeds to the next step.

If the PE-only option is activated ($PE_{flag} = \text{'true.'}$), then the minimum height for the PE calculation region z_{test} is set equal to ht_{lim} . If the PE-only option is not activated ($PE_{flag} = \text{'false.'}$), then output height arrays used in FE calculations are computed:

$$\begin{aligned} zoutma_i &= zout_i - ant_{ref} \\ zoutpa_i &= zout_i - y_{fref} + ant_{ref} \end{aligned} ; \quad i=1,2,...,n_{zout} .$$

The limiting grazing angle, ψ_{lim} , is computed as

$$\psi_{lim} = \mathbf{MAX} \left(.002, \frac{.04443}{f_{MHz}^{.3333}} \right).$$

If more than one refractivity profile has been specified ($n_{prof} > 1$), then ψ_{lim} is multiplied by 2. It is then adjusted for trapping effects by

$$\psi_{lim} = \psi_{lim} + \sqrt{2(rm_{max} - rm_{min})} ,$$

where rm_{max} and rm_{min} are determined in the REFINIT SU. The RO elevation angle limit, α_{lim} , is given by

$$\alpha_{lim} = \sqrt{\psi_{lim}^2 + 2(rm_{tx} - refdum_0)} ,$$

where the variable rm_{tx} and array $refdum$ are determined in the REFINIT SU. Next, the height tolerance, z_{tol} is initialized to .05, and the range and index variables for the RO region, i_{ROp} and x_{ROn} , are initialized to -1 and 0, respectively. The minimum height for the PE calculation region is determined next. The minimum height encompassing all trapping refractive layers is given by

$$h_{test} = h_{trap} + h_{thick} ,$$

where h_{trap} and h_{thick} are determined in the REFINIT SU. If running in full hybrid mode ($i_{hybrid}=1$), the minimum height for the PE calculation region is given by

$$z_{test} = \mathbf{MAX}(h_{test}, 1.2 h_{termax}) ,$$

where h_{termax} is determined in the TERINIT SU. If running in either the partial hybrid mode or PE-only mode, z_{test} is then given by

$$z_{test} = \mathbf{AMAX}(ht_{lim}, ant_{ref}) .$$

The tangent angle, a_{test} , used for automatic calculation of the maximum propagation angle is given by

$$a_{test} = \text{TAN}^{-1} \left(\frac{z_{test} + ant_{ref} + \frac{r_{max}^2}{2a_{ekst}}}{r_{max}} \right),$$

with α_{lim} now set equal to the greater of a_{test} or the previously determined α_{lim} and a_{ekst} is $\frac{4}{3}$ times the mean earth's radius. The GET_K SU is then referenced to determine the effective earth's radius factor.

If rough surface calculations are required ($ruf = \text{'true'}$), then the wavelength, λ , and the free-space wave number, k_o , are initialized using a fixed frequency of 10 GHz (f_{rqg}):

$$\lambda = \frac{c_o}{f_{rqg}}, \quad k_o = \frac{2\pi}{\lambda}.$$

The maximum PE calculation angle is set equal to the greater of 4° ($thmxg$) or the maximum terrain tangent angle, α_u , determined in TERINIT SU. The FFTPAR SU is then referenced to determine PE grid variables and the transform size required. Next, the PEINIT SU is referenced to initialize all arrays and variables associated with the PE calculation algorithm. Variables needed for spectral estimation of the grazing angles are then initialized. The number of bins, n_p , considered in the near-surface PE region, is set equal to 16. The power of 2 transform, ln_p , is set equal to 9. The following variables used in the spectral estimation calculations are given by

$$\begin{aligned} n_s &= 2^{ln_p}, \\ n_{p4} &= \frac{n_p}{4}, \\ n_{p34} &= 3n_{p4}, \\ cn_{p75} &= \frac{\pi}{n_{p4}}. \end{aligned}$$

The ALLARRAY_XORUF SU is then referenced to allocate and initialize all arrays associated with rough surface calculations. The filter array $filtp$ is now determined by

$$filtp_i = \frac{1}{2} + \frac{1}{2} \text{COS}(i \, cn_{p75}), \quad i = 0, 1, 2, \dots, n_{p4},$$

and the variable xo_{con} is given by

$$xo_{con} = \frac{\lambda}{2 n_s \Delta z_{PE}} .$$

The GETGRAZE SU is next referenced to determine all grazing angles for subsequent rough surface calculations. This involves computing a complete PE run out to the maximum range r_{max} . In doing this, the refractivity arrays initially set in the REFINIT SU must be re-initialized for the second, or “real,” PE run that includes rough surface effects.

If rough surface calculations are not required ($ruf = \text{'false.'}$), then the grazing angle array Ψ is initialized to 0, with the number of grazing angles, i_{grz} , set to 1.

Next, the wavelength and free-space wave number are recomputed for the specified frequency f_{MHz} :

$$\lambda = \frac{c_o}{f_{MHz}} , \quad k_o = \frac{2\pi}{\lambda} .$$

The DIEINIT SU is then referenced to initialize all dielectric ground constants. If rough surface calculations are required two terms used in the computation of the rough surface reflection coefficient are determined as follows:

$$ruf_{fac} = \frac{4 \pi (.0051)}{\lambda} ,$$

$$ruf_{ht} = ruf_{fac} \, wind_1^2$$

where $wind_1$ is the first wind speed in m/s provided by the calling CSCI.

If the PE-only option is activated ($PE_{flag} = \text{'true.'}$), then the maximum propagation calculation angle used in the PE region, Θ_{max} , is determined according to:

$$\Theta_{max} = \frac{4}{3} th_{max} .$$

The minimum transform size is then set to ln_{min} plus 1 for every 5° in Θ_{max} . The FFTPAR SU is then referenced to determine the PE grid variables, and the maximum valid height within the PE calculation region, z_{lim} , is set equal to the smaller of z_{test} and ht_{lim} . The PEINIT SU is then referenced to initialize all PE variables and arrays and, if required (i.e., $i_{alg} > 0$), the ALN_INIT SU is referenced to initialize all arrays and variables used in the surface impedance calculations.

If the PE-only option is not activated ($PE_{flag} = \text{'false.'}$) and if using the airborne hybrid mode ($i_{hybrid}=0$), then the maximum PE calculation angle is determined from Θ_{75} , obtained from the GET_K SU, according to

$$\Theta_{max} = \frac{4}{3} \Theta_{75} ,$$

and the minimum transform size is set to ln_{min} plus 1 for every 5° in Θ_{max} . The FFTPAR SU is then referenced to determine the PE grid variables, and the maximum valid height within the PE calculation region, z_{lim} , is set equal to z_{test} . If the airborne mode is not the mode of execution ($i_{hybrid} \neq 0$), then the GETTHMAX SU is referenced to determine the minimum angle Θ_{75} to use within the PE calculation region. z_{lim} is then set equal to the smaller of z_{lim} and ht_{lim} .

In order to determine if XO calculations are required, the following steps 1 through 6 are performed for i_{hybrid} not equal to 0.

1. If z_{lim} is less than $ht_{lim} \cdot 10^{-5}$, then the SU proceeds with steps 2 through 6. Otherwise, these steps are skipped and the output range and index, r_{atz} and i_{ratz} , respectively, are calculated as

$$\begin{aligned} r_{atz} &= 2 r_{max} , \\ i_{ratz} &= n_{rout} + 1 \end{aligned}$$

2. The bin number jz_{lim} , corresponding to z_{lim} , is given by

$$jz_{lim} = \text{INT} \left(\frac{z_{lim}}{\Delta z_{PE}} \right) ,$$

and z_{lim} is recomputed such that it corresponds to an integer multiple of bins or mesh heights: $z_{lim} = jz_{lim} \Delta z_{PE}$.

3. Next, r_{atz} and i_{ratz} are determined based on the height, angle, and range arrays $htemp$, $raya$, and $rtemp$, previously determined in the GETTHMAX SU. First, the index j is initialized to i_{ap} (previously determined in the GETTMAX SU) and the index id is initialized to 1. The following steps 3.a through 3.b are repeated until j is greater than i_{rtemp} .
 - a. If $htemp_j$ is greater than z_{lim} , then the iteration is ended and the SU proceeds with step 4.

- b. If $htemp_j$ is greater than zrt_{id} , then id is incremented by 1. The index j is now incremented by 1. Steps 3a through 3b are then repeated.
4. The index ira is set equal to the greater of 1 or $j-1$; the index idg is set equal to $id-1$; and the gradient g_{rd} is set equal to gr_{idg} .
5. Next, the ray with initial launch angle a_{launch} is traced from height $htemp_{ira}$ to z_{lim} . The square of the local ray angle, rad , at the end of the ray trace step is given by

$$rad = raya_{ira}^2 + 2 g_{rd} (z_{lim} - htemp_{ira}).$$

The local ray angle, a_{atz} , at height z_{lim} is initialized to 0. If rad is greater than 0, then a_{atz} is given by

$$a_{atz} = \text{SIGN}(1, raya_{ira}) \sqrt{rad},$$

and the range r_{atz} is now given by

$$r_{atz} = rtemp_{ira} + \frac{a_{atz} - raya_{ira}}{g_{rd}}.$$

6. If r_{atz} is less than r_{max} and z_{lim} is less than ht_{lim} , then the index k is determined such that $rngout_k > r_{atz}$ and $rngout_{k-1} < r_{atz}$. Then i_{ratz} is set equal to the smaller of n_{rout} and k , and i_{xostp} is set equal to i_{ratz} . The number of XO calculations needed, i_{xo} , is then set equal to i_{xostp} .

The PEINIT SU is next referenced to initialize all variables and arrays necessary for PE calculations. All variables and arrays associated with XO calculations are now initialized, provided i_{xo} is greater than 0. The maximum number of points, iz_{max} , allocated for arrays used in XO calculations, is determined by

$$iz_{max} = \frac{\text{NINT}\left(\frac{r_{max} - r_{atz}}{\Delta r_{PE}}\right)}{iz_{inc}} + 4.$$

Next, variables needed for spectral estimation calculations are initialized. The number of bins, n_p , considered in the upper PE region, is set equal to 8 if no terrain profile or wind speeds are specified, and 16, otherwise. The power of 2 transform, ln_p , is set equal to 8. The following variables are given by

$$\begin{aligned}
n_s &= 2^{ln_p}, \\
n_{p4} &= \frac{n_p}{4}, \\
n_{p34} &= 3n_{p4}, \\
cn_{p75} &= \frac{\pi}{n_{p4}}.
\end{aligned}$$

The ALLARRAY_XORUF SU is then referenced to allocate and initialize all arrays associated with XO calculations. The filter array $filtp$ is now determined by

$$filtp_i = \frac{1}{2} + \frac{1}{2} \cos(i cn_{p75}), \quad i = 0, 1, 2, \dots, n_{p4},$$

and the variable xo_{con} is given by

$$xo_{con} = \frac{\lambda}{2 n_s \Delta z_{PE}}$$

The ALN_INIT SU is next referenced to initialize all arrays and variables used in the surface impedance calculations, and the FILLHT SU is referenced to obtain the $htfe$ array separating the FE from the RO region.

Several variables associated with the beginning, middle, and end of a PE range step, are initialized. If rough surface calculations are required, the GRAZE_INT SU is referenced to interpolate the pre-computed grazing angles (as determined in the GETGRAZE SU) at every output range step. If troposcatter calculations are required ($T_{ropo} = 1$), then the TROPOINT SU is referenced to initialize all variables and arrays. An additive loss term pl_{cnst} and the free-space loss fsl are determined at each output range step by

$$\begin{aligned}
pl_{cnst} &= 20 \text{LOG}_{10}(2 k_o), \\
fsl_i &= rlog o_i + pl_{cnst} ; \quad i = 1, 2, \dots, n_{rout}.
\end{aligned}$$

Finally, if the absorption flag k_{abs} is equal to 1, then the GASABS SU is referenced to determine the absorption attenuation rate, gas_{att} . If k_{abs} is equal to 2, then gas_{att} is determined by the calling CSCI-specified attenuation rate, γ_a , multiplied by 10^{-3} to convert γ_a from dB/km to dB/m. Several dynamically allocated terrain arrays used in the TERINIT SU are deallocated and the CSC is exited.

Table 8 and Table 9 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the APMINIT CSC.

Table 8. APMINIT CSC input data element requirements.

Name	Description	Units	Source
abs_{hum}	Absolute humidity near the surface	g/meters ³	Calling CSCI
a_{ekst}	$\frac{4}{3}$ times mean earth radius	meters	APM_MOD
ant_{ht}	Transmitting antenna height above local ground	meters	Calling CSCI
c_o	Speed of light multiplied by 10^{-6}	meters/sec	APM_MOD
$dielec$	2-dimensional array containing the relative permittivity and conductivity; $dielec_{1,i}$ and $dielec_{2,i}$, respectively.	N/A, S/m	Calling CSCI
f_{MHz}	Frequency	MHz	Calling CSCI
f_{rqg}	Frequency in MHz at which to perform grazing angle calculations	MHz	APMINIT CSC
γ_a	Gaseous absorption attenuation rate	dB/km	Calling CSCI
$hfang$	Cut-back angles	degrees	Calling CSCI
$hffac$	Cut-back antenna pattern factors	N/A	Calling CSCI
h_{max}	Maximum output height with respect to mean sea level	meters	Calling CSCI
h_{min}	Minimum output height with respect to mean sea level	meters	Calling CSCI
$hmsl$	2-dimensional array containing heights with respect to mean sea level of each profile. Array format must be $hmsl_{i,j}$ = height of i^{th} level of j^{th} profile; $j=1$ for range-independent cases	meters	Calling CSCI
i_{extra}	Extrapolation flag for refractivity profiles entered in combination with terrain below mean sea level 0 = extrapolate to minimum terrain height standard atmosphere gradient 1 = extrapolate to minimum terrain height using first gradient in profile	N/A	Calling CSCI
i_{gr}	Number of different ground types specified	N/A	Calling CSCI
$igrnd$	Integer array containing ground type composition for given terrain profile - can vary with range. Different ground types are: 0 = sea water 1 = fresh water 2 = wet ground 3 = medium dry ground 4 = very dry ground 5 = ice at -1 degree C 6 = ice at -10 degree C 7 = user defined (in which case, values of relative permittivity and conductivity must be given).	N/A	Calling CSCI

Table 8. APMINIT CSC input data element requirements. (Continued)

Name	Description	Units	Source
i_{pat}	Antenna pattern type 1 = Omni-directional 2 = Gaussian 3 = Sine(x)/x 4 = Cosecant-squared 5 = Generic height-finder 6 = User-defined height-finder 7 = User-defined antenna patter	N/A	Calling CSCI
i_{pol}	Polarization flag: 0 = horizontal polarization 1 = vertical polarization	N/A	Calling CSCI
i_{temp}	Temporary number of range steps (used for ray tracing)	N/A	APM_MOD
i_{tp}	Number of height/range points in profile	N/A	Calling CSCI
$lerr6$	Logical flag to allow for error -6 to be bypassed	N/A	Calling CSCI
$lerr12$	Logical flag to allow for error -12 to be bypassed	N/A	Calling CSCI
$lvlp$	Number of levels in refractivity profile	N/A	Calling CSCI
n_{fac}	Number of user-defined cut-back angles and cut-back antenna factors for user-specified height-finder antenna type	N/A	Calling CSCI
n_{prof}	Number of refractivity profiles	N/A	Calling CSCI
n_{rout}	Number of output height points desired	N/A	Calling CSCI
n_w	Number of wind speeds	N/A	Calling CSCI
n_{zout}	Number of output range points desired	N/A	Calling CSCI
PE_{flag}	Flag to indicate use of PE algorithm only: ‘.true.’ = only use PE sub-model ‘.false.’ = use automatic hybrid model	N/A	Calling CSCI
π	Constant equal to the value of π	N/A	APM_MOD
$refmsl$	2-dimensional array containing refractivity with respect to mean sea level of each profile. Array format must be $refmsl_{i,j}$ = M-unit at i^{th} level of j^{th} profile; $j=1$ for range-independent cases	M-units	Calling CSCI
$rgrnd$	Array containing ranges at which varying ground types apply.	meters	Calling CSCI
r_{max}	Maximum specified range	meters	Calling CSCI
r_{mult}	PE range step multiplication factor	N/A	Calling CSCI
$rngprof$	Ranges of each profile: $rngprof_i$ = range of i^{th} profile	meters	Calling CSCI
$rngwind$	Ranges of wind speeds entered: $rngwind_i$ = range of i^{th} wind speed	meters	Calling CSCI
t_{air}	Air temperature near the surface	°C	Calling CSCI
$terx$	Range points of terrain profile	meters	Calling CSCI
$tery$	Height points of terrain profile	meters	Calling CSCI

Table 8. APMINIT CSC input data element requirements. (Continued)

Name	Description	Units	Source
th_{max}	Visible portion of maximum PE calculation angle	degrees	Calling CSCI
ϑ_{mxg}	Maximum PE calculation angle for spectral estimation of grazing angles	radians	APMINIT CSC
T_{ropo}	Troposcatter calculation flag: 'false.' = no troposcatter calcs 'true.' = troposcatter calcs	N/A	Calling CSCI
μ_{bw}	Antenna vertical beamwidth	degrees	Calling CSCI
μ_o	Antenna elevation angle	degrees	Calling CSCI
$wind$	Array of wind speeds	meters/sec	Calling CSCI

Table 9. APMINIT CSC output data element requirements.

Name	Description	Units
a_{atz}	Local ray or propagation angle at height z_{lim} and range r_{atz}	radians
a_{launch}	Launch angle used which, when traced, separates PE and XO regions from the RO region	N/A
α_{lim}	Elevation angle of the RO limiting ray	radians
ant_{fac}	Antenna pattern parameter (depends on i_{pat} and μ_{bw})	N/A
Δr_{grz}	PE range step used for calculation of grazing angles	meters
Δr_{out}	Output range step	meters
Δz_{out}	Output height increment	meters
$dielec$	2-dimensional array containing the relative permittivity and conductivity; dielec1,i and dielec2,i, respectively.	N/A, S/m
$filtp$	Array filter for spectral estimation calculations	N/A
fsl	Free space loss array for output ranges	dB
gas_{att}	Gaseous absorption attenuation rate	dB/m
$hfangr$	Cut-back angles	radians
i_{alg}	Integer flag indicating which DMFT algorithm is being used: 0 = no DMFT algorithm will be used 1 = use central difference algorithm 2 = use backward difference algorithm	N/A
i_{error}	Error flag	N/A
i_g	Counter indicating current ground type being modeled	N/A
i_{hybrid}	Integer indicating which sub-models will be used: 0 = pure PE model 1 = full hybrid model (PE + FE + RO + XO) 2 = partial hybrid model (PE + XO)	N/A
i_o	Starting index for $mpfl$ array: 0 = 1 st calculated output point is at surface 1 = 1 st calculated output point is at height Δz_{out}	N/A
i_{PE}	Number of PE range steps	N/A

Table 9. APMINIT CSC output data element requirements. (Continued)

Name	Description	Units
i_{ratz}	Index of output range step in which to begin storing propagation factor and outgoing angle for XO region	N/A
i_{ROp}	Array index for previous range in RO region	N/A
i_{tpa}	Number of height/range points pairs in terrain profile arrays tx , ty	N/A
i_{xo}	Number of range steps in XO calculation region	N/A
i_{xostp}	Current output range step index for XO calculations	N/A
iz	Number of propagation factor, range, angle triplets stored in array $ffacz$	N/A
iz_{max}	Maximum number of points allocated for arrays associated with XO calculations	N/A
jz_{lim}	PE bin # corresponding to z_{lim} , i.e., $z_{lim} = jz_{lim} \Delta z_{PE}$	N/A
k_{abs}	Gaseous absorption calculation flag: 0 = no absorption loss 1 = compute absorption loss based on air temperature t_{air} and absolute humidity abs_{hum} 2 = compute absorption loss based on specified absorption attenuation rate γ_a	N/A
k_o	Free-space wave number	meters ⁻¹
λ	Wavelength	meters
ln_{min}	Minimum power of 2 transform size	N/A
ln_p	Power of 2 transform size used in spectral estimation calculations; i.e., $n_p = 2^{ln_p}$	N/A
μ_{or}	Antenna pattern elevation angle	radians
μ_{bwr}	Antenna vertical beamwidth in radians	radians
μ_{max}	Limiting angle for SIN(X)/X and generic height-finder antenna pattern factors	N/A
n_p	Number of bins in upper and near-surface PE region to consider for spectral estimation.	N/A
n_{p34}	$\frac{3}{4} n_p$	N/A
n_{p4}	$\frac{1}{4} n_p$	N/A
n_s	Transform size for spectral estimation calculations	N/A
p_{elev}	Sine of antenna elevation angle	N/A
pl_{cnst}	Constant used in determining propagation loss ($pl_{cnst} = 20 \log_{10}(2 k_o)$)	dB/m
ψ_{lim}	Grazing angle of limiting ray	radians
r_{atz}	Range at which z_{lim} is reached (used for hybrid model)	meters
$rlogo$	Array containing 20 times the logarithm of all output ranges	N/A
$rngout$	Array containing all desired output ranges	meters
$rsqrd$	Array containing the square of all desired output ranges	meters ²
ruf	Logical flag indicating if rough sea surface calculations are required ‘.true.’ = perform rough sea surface calculations ‘.false.’ = do not perform rough sea surface calculations	N/A
ruf_{fac}	Factor used for wave height calculation	meters ⁻¹
s_{bw}	Sine of antenna vertical beam width	N/A

Table 9. APMINIT CSC output data element requirements. (Continued)

Name	Description	Units
θ_{max}	Maximum propagation angle used in PE calculations	radians
θ_{75}	75% of maximum propagation angle in PE calculations	radians
xo_{con}	Constant used in determining outgoing propagation angle ϑ_{out}	N/A
y_{cur}	Height of ground at current range r	meters
y_{curm}	Height of ground midway between last and current PE range	meters
y_{ref}	Ground elevation height at source	meters
y_{last}	Height of ground at previous range r_{last}	meters
$xROn$	Next range in RO region	meters
z_{lim}	Height limit for PE calculation region	meters
$zout$	Array containing all desired output heights referenced to h_{minter}	meters
$zoutma$	Array output heights relative to "real" ant_{ref}	meters
$zoutpa$	Array output heights relative to "image" ant_{ref}	meters
zro	Array of output heights in RO region	meters
z_{test}	Height in PE region that must be reached for hybrid model	meters
z_{tol}	Height tolerance for Newton's method	meters

5.1.1 Allocate Arrays APM (ALLARRAY_APM) SU

The purpose of the ALLARRAY_APM SU is to allocate and initialize all dynamically dimensioned arrays associated with APM terrain, refractivity, troposcatter, and general variable arrays.

The ALLARRAY_APM SU utilizes the FORTRAN ALLOCATE and DEALLOCATE statements to dynamically size arrays that have been previously declared with the ALLOCATABLE attribute in the APM_MOD MODULE or to free the array storage space previously reserved in an ALLOCATE statement. Each dimension of the ALLOCATABLE array is indicated by a colon in the APM_MOD module (e.g., *array(:)*). The ALLOCATE statement establishes the upper and lower bounds of each dimension and reserves sufficient memory. Because attempting to allocate a previously allocated array causes a run-time error, each ALLOCATE statement for an array is preceded by a test to determine if it has been allocated. If it has, it is deallocated first before it is allocated.

Initially, the integer used to indicate an error, i_{error} , is set to zero. If, in attempting to allocate an array, a value of i_{error} other than zero is returned by an ALLOCATE statement, the SU is exited. Note that each array that is dynamically allocated in this SU is initialized to zero.

There are six integers input to this SU that are used to dynamically allocate the arrays. Unless otherwise indicated, these integers are used as the single dimension of the dynamically allocated array. The first, i_{gr} , is the number of different ground types

specified. The second, i_{tpa} , is the number of terrain points used internally in arrays tx and ty . The third, $lvlp$, is the number of levels in the refractivity profile. The fourth, n_{fac} , is the number of user-defined cut-back antenna pattern factors for the user-defined height-finder antenna type. The fifth, n_{rout} , is the integer number of output range points desired. And finally, the sixth, n_{zout} , is the integer number of output height points desired.

The definitions of the following arrays allocated in this SU are given in Table 11. The only array that is allocated using the integer n_{fac} is $hfangr$. The arrays that are allocated using the integer n_{rout} are $rsqrd$, fsl , $rlogo$, $hlim$, $htfe$, and $rngout$. The arrays that are allocated using the integer n_{zout} are $zout$, $zoutma$, $zoutpa$, $rfac1$, $rfac2$, and $rloss$. Only those arrays associated with the hybrid mode of execution will be allocated; i.e., if the PE-only mode is activated ($PE_{flag} = \text{'false'}$), then arrays $rsqrd$, $zoutma$, $zoutpa$, $hlim$, and $htfe$ will not be allocated.

The arrays associated with terrain information use either the integer i_{tpa} or the integer i_{gr} . The arrays that are allocated with the integer i_{tpa} are tx , ty , and slp . The arrays allocated using the integer i_{gr} are $igrnd$, $rgrnd$, and nc^2 . The array $dielec$ is allocated using two as the first dimension and i_{gr} as the second dimension. While arrays $igrnd$, $rgrnd$, and $dielec$ are usually specified by the calling CSCI, it is not necessary when performing an over-water case. In this case, i_{gr} can have a value of 0, and these arrays will be defaulted to size of one element with a sea-surface ground type.

All refractivity arrays used in the PE algorithm are allocated using the integer $lvlp$ and include $refdum$, $htdum$, $grdum$, $href$, and $refref$.

The arrays associated with troposcatter calculations use either integers n_{zout} and n_{rout} and will be allocated only if troposcatter calculations are required for the particular APM CSCI application ($T_{ropo} = \text{'true'}$). The arrays allocated using the integer n_{zout} include $adif$, $d2s$, rdt , and $v2s$, and the array allocated for size n_{rout} is $\varnothing$.

Table 10 and Table 11 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the ALLARRAY_APM SU.

Table 10. ALLARRAY_APM SU input data element requirements.

Name	Description	Units	Source
i_{gr}	Number of different ground types specified	N/A	Calling CSCI
i_{tpa}	Number of terrain points used internally in terrain profile arrays tx , ty	N/A	APMINIT CSC
lv/p	Number of levels in refractivity profile	N/A	Calling CSCI
n_{facs}	Number of user-defined cut-back angles and cut-back antenna factors for user-specified height-finder antenna type	N/A	Calling CSCI
n_{rout}	Number of output height points desired	N/A	Calling CSCI
n_{zout}	Number of output range points desired	N/A	Calling CSCI
PE_{flag}	Flag to indicate use of PE algorithm only: ‘.true.’ = only use PE sub-model ‘.false.’ = use automatic hybrid mode	N/A	Calling CSCI
T_{ropo}	Troposcatter calculation flag: ‘.false.’ = no troposcatter calcs ‘.true.’ = troposcatter calcs	N/A	Calling CSCI

Table 11. ALLARRAY_APM SU output data element requirements.

Name	Description	Units
$adif$	Height differences between ant_{ref} and all output receiver heights	meters
$d2s$	Array of tangent ranges for all output receiver heights over smooth surface	meters
$dielec$	2-dimensional array containing the relative permittivity and conductivity; $dielec_{1,i}$ and $dielec_{2,i}$, respectively.	N/A, S/m
fsl	Free space loss array for output ranges	dB
$grdum$	Array of refractivity gradients defined by profile $htdum$ and $refdum$	M-units/ meter
$hfangr$	Cut-back angles	radians
$hlim$	Array containing height at each output range separating the RO region from the PE (at close ranges) and XO (at far ranges) regions	meters
$href$	Heights of refractivity profile with respect to y_{ref}	meters
$htdum$	Height array for current interpolated profile	meters
$htfe$	Array containing the height at each output range separating the FE region from the RO region (full hybrid mode), or the FE region from the PE region (partial hybrid mode)	meters
i_{error}	Integer variable indicating error number for ALLOCATE and DEALLOCATE statements	N/A

Table 11. ALLARRAY_APM SU output data element requirements. (Continued)

Name	Description	Units
<i>igrnd</i>	Integer array containing ground type composition for given terrain profile - can vary with range. Different ground types are: 0 = sea water 1 = fresh water 2 = wet ground 3 = medium dry ground 4 = very dry ground 5 = ice at -1 degree C 6 = ice at -10 degree C 7 = user defined (in which case, values of relative permittivity and conductivity must be given).	N/A
<i>nc</i> ²	Array of complex dielectric constants	N/A
<i>rdt</i>	Array of minimum ranges at which diffraction field solutions are applicable (for smooth surface) for all output receiver heights.	meters
<i>refdum</i>	M-unit array for current interpolated profile	M-units
<i>refref</i>	Refractivity profile with respect to y_{ref}	M-units
<i>rfac1</i>	Propagation factor at valid output height points from PE field at range r_{last} .	dB
<i>rfac2</i>	Propagation factor at valid output height points from PE field at range r	dB
<i>rgrnd</i>	Array containing ranges at which varying ground types apply	meters
<i>rlogo</i>	Array used for storing 20 LOG ₁₀ (output ranges)	N/A
<i>rloss</i>	Propagation loss	dB
<i>rngout</i>	Array containing all desired output ranges	meters
<i>rsqrd</i>	Array containing the square of all desired output ranges	meters ²
<i>slp</i>	Slope of each segment of terrain	N/A
<i>ϑ0</i>	Array of angles used to determine common volume scattering angle	radians
<i>ϑ2s</i>	Array of tangent angles from all output receiver heights - used with smooth surface	radians
<i>tx</i>	Range points of terrain profile	meters
<i>ty</i>	Adjusted height points of terrain profile	meters
<i>zout</i>	Array containing all desired output heights referenced to h_{minter}	meters
<i>zoutma</i>	Array output heights relative to "real" ant_{ref}	meters
<i>zoutpa</i>	Array output heights relative to "image" ant_{ref}	meters

5.1.2 Allocate Array PE (ALLARRAY_PE) SU

The purpose of the ALLARRAY_PE SU is to allocate and initialize all dynamically dimensioned arrays associated with PE calculations.

The ALLARRAY_PE SU utilizes the FORTRAN ALLOCATE and DEALLOCATE statements to dynamically size arrays that have been previously declared with the ALLOCATABLE attribute in the APM_MOD module or to free the array

storage space previously reserved in an ALLOCATE statement. Each dimension of the ALLOCATABLE array is indicated by a colon in the APM_MOD MODULE (e.g., *array(:)*). The ALLOCATE statement establishes the upper and lower bounds of each dimension and reserves sufficient memory. Because attempting to allocate a previously allocated array causes a run-time error, each ALLOCATE statement for an array is preceded by a test to determine if it has been allocated. If it has, it is deallocated first before it is allocated.

Initially, the integer used to indicate an error, i_{error} , is set to zero. If in attempting to allocate an array, a value of i_{error} other than zero is returned by an ALLOCATE statement, then the SU is exited. Note that each array that is dynamically allocated in this SU is initialized to zero.

There are two integers input to this SU that are used to dynamically allocate the arrays. Unless otherwise indicated, these integers are used as the single dimension of the dynamically allocated array. The first, n_{fft} , is the transform size. The second, n_4 , is the transform size n_{fft} divided by four.

The definitions of the following arrays allocated in this SU are given in Table 13. The arrays that are allocated using the integer n_{fft} are *envpr*, *frsp*, *U*, *Ulast*, *ht*, *profint*, *udum*, *rn*, *w*, and *ym*. The only array allocated using the integer n_4 is *filt*. If rough sea surface or finite conductivity calculations are not required (*ruf* = '.false.' and $i_{pol} = 0$), then arrays *rn*, *w*, and *ym* will not be allocated.

Table 12 and Table 13 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the ALLARRAY_PE SU.

Table 12. ALLARRAY_PE SU input data element requirements.

Name	Description	Units	Source
i_{pol}	Polarization flag: 0 = horizontal polarization 1 = vertical polarization	N/A	Calling CSCI
n_{fft}	Transform size	N/A	FFTPAR SU
n_4	$\frac{1}{4} n_{fft}$	N/A	APMINIT CSC
<i>ruf</i>	Logical flag indicating if rough sea surface calculations are required 'true.' = perform rough sea surface calculations 'false.' = do not perform rough sea surface calculations	N/A	APMINIT CSC

Table 13. ALLARRAY_PE SU output data element requirements.

Name	Description	Units
<i>envpr</i>	Complex [refractivity] phase term array interpolated every Δz_{PE} in height	N/A
<i>filt</i>	Cosine-tapered (Tukey) filter array	N/A
<i>frsp</i>	Complex free-space propagator term array	N/A
<i>ht</i>	PE mesh height array of size $n_{\#}$	meters
<i>i_{error}</i>	Integer variable indicating error number for ALLOCATE and DEALLOCATE statements	N/A
<i>profint</i>	Profile interpolated to every Δz_{PE} in height	M-units
<i>rn</i>	Array of R_T to the i^{th} power (e.g., $rn_i = R_T^i$)	N/A
<i>U</i>	Complex field at current PE range r	$\mu V/m$
<i>Udum</i>	Dummy array used for temporary storage of real or imaginary part of complex PE field array U	$\mu V/m$
<i>Ulast</i>	Complex field at previous PE range r_{last}	$\mu V/m$
<i>w</i>	Difference equation of complex PE field	$\mu V/m$
<i>ym</i>	Particular solution of difference equation	N/A

5.1.3 Allocate Array RO (ALLARRAY_RO) SU

The purpose of the ALLARRAY_RO SU is to allocate and initialize all dynamically dimensioned arrays associated with RO calculations.

The ALLARRAY_RO SU utilizes the FORTRAN ALLOCATE and DEALLOCATE statements to dynamically size arrays that have been previously declared with the ALLOCATABLE attribute in the APM_MOD MODULE or to free the array storage space previously reserved in an ALLOCATE statement. Each dimension of the ALLOCATABLE array is indicated by a colon in the APM_MOD module (e.g., *array(:)*). The ALLOCATE statement establishes the upper and lower bounds of each dimension and reserves sufficient memory. Because attempting to allocate a previously allocated array causes a run-time error, each ALLOCATE statement for an array is preceded by a test to determine if it has been allocated. If it has, it is deallocated first before it is allocated.

Initially, the integer used to indicate an error, i_{error} , is set to zero. If in attempting to allocate an array, a value of i_{error} other than zero is returned by an ALLOCATE statement, then the SU is exited. Note that each array that is dynamically allocated in this SU is initialized to zero.

There are two integers input to this SU that are used to dynamically allocate the arrays. Unless otherwise indicated, these integers are used as the single dimension of the dynamically allocated array. The first of these is *lvlp*, the number of points in the

refractivity profile, and the second is n_{zout} , which is the number of output height points desired.

The definitions of the following arrays allocated in this SU are given in Table 15. The array *zro* is allocated using the integer n_{zout} . The remainder of the arrays associated with refractivity information, *gr*, *q*, *rm*, and *zrt*, are allocated using the integer *lvlpt*, which is equal to *lvlp* plus one.

Table 14 and Table 15 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the ALLARRAY_RO SU.

Table 14. ALLARRAY_RO input data element requirements.

Name	Description	Units	Source
<i>lvlp</i>	Number of levels in refractivity profile	N/A	Calling CSCI
n_{zout}	Number of desired output height points	N/A	Calling CSCI

Table 15. ALLARRAY_RO output data element requirements.

Name	Description	Units
<i>gr</i>	Intermediate M-unit gradient array	M-units/meter
<i>q</i>	Intermediate M-unit difference array	M-units
<i>rm</i>	Intermediate M-unit array	M-units
<i>zro</i>	Array of output heights	meters
<i>zrt</i>	Intermediate height array	meters

5.1.4 Allocate Array XORUF (ALLARRAY_XORUF) SU

The purpose of the ALLARRAY_XORUF SU is to allocate and initialize all dynamically dimensioned arrays associated with XO and rough sea surface calculations.

The ALLARRAY_XORUF SU utilizes the FORTRAN ALLOCATE and DEALLOCATE statements to dynamically size arrays that have been previously declared with the ALLOCATABLE attribute in the APM_MOD MODULE or to free the array storage space previously reserved in an ALLOCATE statement. Each dimension of the ALLOCATABLE array is indicated by a colon in the APM_MOD module (e.g., *array(:)*). The ALLOCATE statement establishes the upper and lower bounds of each dimension and reserves sufficient memory. Because attempting to allocate a previously allocated array causes a run-time error, each ALLOCATE statement for an array is preceded by a test to determine if it has been allocated. If it has, it is deallocated first before it is allocated.

Initially, the integer used to indicate an error, i_{error} , is set to zero. If in attempting to allocate an array, a value of i_{error} other than zero is returned by an ALLOCATE statement, then the SU is exited. Note that each array that is dynamically allocated in this SU is initialized to zero.

There are six integers input to this SU that are used to dynamically allocate the arrays. Unless otherwise indicated, these integers are used as the single dimension of the dynamically allocated array. The first of these is i_{xo} and is the number of XO range step calculations required. The second is iz_{max} , the maximum number of points allocated for arrays associated with XO calculations. The third is $lvlp$, the number of points in the refractivity profile. The fourth is n_{p4} , 1/4 of the number of points, n_p , used in the top or bottom portion of the PE region for spectral estimation. The fifth is n_{rout} , the number of output range points desired, and the last is n_s , the transform size used in spectral estimation calculations (i.e., $n_s = 2^{ln_p}$), where the integer ln_p is the power of 2 transform size.

The definitions of the following arrays allocated in this SU are given in Table 17. The array *ffrout* is allocated using the integer n_{rout} as the first dimension and two as the second dimension. The integer iz_{max} is used as the first dimension limit, and the integer three is used as the second dimension in the allocation of the array *ffacz*. Both of the arrays, *grad* and *htr* are allocated with the first dimension given by $lvlp$ and the second dimension given by iz_{max} . The array *lvl* is allocated using the integer iz_{max} . The array *filtp* is allocated using the integer n_{p4} . The three arrays *xp*, *yp*, and *spectr* are allocated using the integer n_s .

If no XO calculations are required ($i_{xo} = 0$), then the arrays *ffrout*, *ffacz*, *grad*, *htr*, and *lvl* will not be allocated.

Table 16 and Table 17 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the ALLARRAY_XORUF SU.

Table 16. ALLARRAY_XORUF SU input data element requirements.

Name	Description	Units	Source
i_{xo}	Number of range steps in XO calculation region	N/A	APMINIT CSC
iz_{max}	Maximum number of points allocated for arrays associated with XO calculations	N/A	APMINIT CSC
$lvlp$	Number of height/refractivity levels in profiles	N/A	Calling CSCI
n_{p4}	$\frac{1}{4} n_p$	N/A	APMINIT CSC
n_{rout}	Integer number of output range points desired	N/A	Calling CSCI
n_s	Transform size for spectral estimation calculations	N/A	APMINIT CSC

Table 17. ALLARRAY_XORUF SU output data element requirements.

Name	Description	Units
<i>ffacz</i>	2-dimensional array containing propagation factor, range, and outgoing propagation angle at z_{lim}	dB, meters, radians
<i>ffrout</i>	2-dimensional array of propagation factors at each output range beyond r_{atz} and at height z_{lim}	dB
<i>filtp</i>	Array filter for spectral estimation calculations	N/A
<i>grad</i>	2-dimensional array containing gradients of each profile used in XO calculations	M-units/ meter
<i>htr</i>	2-dimensional array containing heights of each profile used in XO calculations	meters
<i>i_{error}</i>	Integer variable indicating error number for ALLOCATE and DEALLOCATE statements	N/A
<i>lvl</i>	Number of height levels in each profile used in XO calculations	N/A
<i>spectr</i>	Spectral amplitude of field	dB
<i>xp</i>	Real part of spectral portion of PE field	$\mu\text{V/m}$
<i>yp</i>	Imaginary part of spectral portion field	$\mu\text{V/m}$

5.1.5 Alpha Impedance Initialization (ALN_INIT) SU

The purpose of the ALN_INIT SU is to initialize variables and arrays used for either the backward or central difference form of the DMFT algorithm. The DMFT algorithm is used only for finite conductivity and/or rough sea surface calculations.

Upon entering, the GETALN SU is referenced to obtain the surface impedance, α_s , the complex root R_T , the array rn containing the powers of the complex root, and a coefficient R_k used in the central difference algorithm. These variables are computed from the grazing angle ψ and current range, which, for initialization purposes are set equal to $\pi/2$ and 0, respectively.

If the central difference algorithm is required ($i_{alg}=1$) for the particular APM CSCI application, then coefficients necessary for this form of the DMFT are computed as follows:

$$ck_1 = R_k \left[.5(U_0 + U_{n_{ffl}} rn_{n_{ffl}}) + \sum_{i=1}^{n_{ml}} U_i rn_i \right]$$

$$ck_2 = R_k \left[.5(U_0 rn_{n_{ffl}} + U_{n_{ffl}}) + \sum_{i=1}^{n_{ml}} (-1)^i U_{n_{ffl}-i} rn_i \right]$$

If the backward difference algorithm is required ($i_{alg}=2$) for the particular APM CSCI application, then the variable $cmft$ is initialized using the starting PE field U and the array rn according to

$$cmft = \sum_{i=1}^{n_{m1}} U_i rn_i .$$

Table 18 and Table 19 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the ALN_INIT SU.

Table 18. ALN_INIT SU input data element requirements.

Name	Description	Units	Source
ψ	Grazing angle	radians	APMINIT CSC
i_{alg}	Integer flag indicating which DMFT algorithm is being used: 0 = no DMFT algorithm will be used 1 = use central difference algorithm 2 = use backward difference algorithm	N/A	APMINIT CSC
r	Current range	meters	APMINIT CSC
n_{m1}	$n_{fft}-1$	N/A	APMINIT CSC
R_k	Constant used to compute coefficients in central difference form of the DMFT	N/A	GETALN SU
rn	Array of R_T to the i^{th} power (e.g., $rn_i = R_T^i$)	N/A	GETALN SU
U	Complex field at current PE range r	$\mu V/m$	XYINIT SU

Table 19. ALN_INIT SU output data element requirements.

Name	Description	Units
ck_1	Coefficient used in central difference form of DMFT	N/A
ck_2	Coefficient used in central difference form of DMFT	N/A
$cmft$	Coefficient used in backward difference form of DMFT	N/A

5.1.6 Antenna Pattern (ANTPAT) SU

The purpose of the ANTPAT SU is to calculate an antenna pattern factor (normalized antenna gain), $f(\alpha)$, for a specified antenna elevation angle, α . Currently, antenna pattern factors are included for six types of antennas. These patterns include an omni-directional ($i_{pat}=1$) type; a Gaussian ($i_{pat}=2$) type; a Sin(X)/X ($i_{pat}=3$) type; a cosecant-squared ($i_{pat}=4$) type; a generic height-finder ($i_{pat}=5$) type; a user-defined height-finder type ($i_{pat}=6$), in which the calling CSCI must also specify an array of cut-back

angles and pattern factors; and finally, a user-defined antenna type ($i_{pat}=7$), in which the calling CSCI must specify an array of antenna pattern factors and angles.

From two antenna pattern parameters, ant_{fac} and p_{elev} ; the antenna beam width, μ_{bwr} , and elevation angle, μ_{or} ; a specified angle, α , for which the antenna pattern factor is desired; and the antenna radiation pattern type, i_{pat} , the antenna factor is calculated as follows.

If the antenna pattern is omni-directional, then $f(\alpha) = 1$. If the antenna pattern is Gaussian, then

$$f(\alpha) = e^{-ant_{fac} [\text{SIN}(\alpha) - p_{elev}]^2}.$$

If the antenna pattern is cosecant-squared, compute the elevation angle relative to the antenna elevation angle as

$$\alpha_{pat} = \alpha - \mu_{or}.$$

The antenna pattern is now given as

$$f(\alpha) = \frac{s_{bw}}{\text{SIN}(\alpha_{pat})}, \quad \text{for } \alpha_{pat} > \mu_{bwr},$$

$$f(\alpha) = \text{MAX} \left(.03, \left[1 + \frac{\alpha_{pat}}{\mu_{bwr}} \right] \right), \quad \text{for } \alpha_{pat} < 0,$$

$$f(\alpha) = 1 \quad \text{otherwise,}$$

where s_{bw} is determined in the APMINIT CSC.

If the antenna pattern is Sin(X)/X, a generic height-finder, or a user-specified height-finder, the following calculations are made.

1. The elevation angle relative to the antenna elevation angle, α_{pat} , is determined as in the previous definition. If the antenna radiation pattern type is a generic or user-specified height-finder, the radiation pattern is simulated as a Sin(X)/X type with the elevation angle adjusted to account for the current pointing angle of the main beam, χ . χ is set equal to the antenna elevation angle μ_{or} . If the direct-path ray angle, α_d , is

greater than the antenna elevation angle, then α_{pat} is computed as $\alpha_{pat} = \alpha - \alpha_d$ and χ is set equal to α_d .

2. The antenna pattern is now given as

$$f(\alpha) = 1 \quad \text{for } |\alpha_{pat}| \leq 10^{-6}$$

$$f(\alpha) = \frac{\text{SIN}[ant_{fac} \text{SIN}(\alpha_{pat})]}{ant_{fac} \text{SIN}(\alpha_{pat})}; \quad \text{for } |\alpha_{pat}| < \mu_{max},$$

$$f(\alpha) = .03, \text{ otherwise.}$$

For a user-defined height-finder, the pattern factor is further adjusted by a power reduction factor, $hffac_i$, as

$$f(\alpha) = f(\alpha) hffac_i; \quad i = n_{fac}, \dots, 2, 1$$

where i is an angle counter, decremented by one from the number of power reduction angles, n_{fac} , for each power reduction angle, $hfangr_i$, which exceeds χ .

For the user-defined antenna type, the antenna pattern factor is simply the array $hffac$ provided by the calling CSCI at angles $hfangr$, where pattern factors are interpolated from $hffac$ for angles α within $hfangr$.

Table 20 and Table 21 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the ANTPAT SU.

Table 20. ANTPAT SU input data element requirements.

Name	Description	Units	Source
ant_{fac}	Antenna pattern parameter (depends on i_{pat} and μ_{bwr})	N/A	APMINIT CSC
α	Antenna elevation angle	radians	Calling SU
α_d	Direct ray elevation angle	radians	AIRBORNE SU FEM SU ROCALC SU TROPOINT SU TROPOSCAT SU XYINIT SU
$hfangr$	Cut-back angles if $i_{pat} = 6$; Antenna pattern angles if $i_{pat} = 7$	radians	Calling CSCI
$hffac$	Cut-back antenna pattern factors if $i_{pat} = 6$; Antenna pattern factors if $i_{pat} = 7$	N/A	Calling CSCI
i_{pat}	Antenna pattern type 1 = Omni-directional 2 = Gaussian 3 = Sine(x)/x 4 = Cosecant-squared 5 = Generic height-finder 6 = User-defined height-finder 7 = User-defined antenna patter	N/A	Calling CSCI
μ_{or}	Antenna pattern (pointing) elevation angle	radians	APMINIT CSC
μ_{bwr}	Antenna vertical beam width	radians	APMINIT CSC
μ_{max}	Limiting angle for Sin(X)/X and generic height-finder antenna pattern factors	radians	APMINIT CSC
n_{facs}	Number of user-defined cut-back angles and cut-back pattern factors	N/A	Calling CSCI
p_{elev}	Sine of antenna elevation angle	N/A	APMINIT CSC
s_{bw}	Sine of antenna vertical beam width	N/A	APMINIT CSC

Table 21. ANTPAT SU output data element requirements.

Name	Description	Units
$f(\alpha)$	Antenna pattern factor for elevation angle α	N/A

5.1.7 Dielectric Initialization (DIEINIT) SU

The purpose of the DIEINIT SU is to determine the conductivity and relative permittivity as a function of frequency in MHz based on general ground composition types.

The DIEINIT SU supports the following general ground types: salt water, fresh water, wet ground, medium dry ground, very dry ground, ice at -1°C , ice at -10°C , and user-defined. For all ground types other than “user-defined,” the permittivity and conductivity are calculated as functions of frequency from curve fits to the permittivity and conductivity graphs shown in the Recommendations and Reports of the International Radio Consultative Committee (Ref. 4). For the i^{th} input ground type case, $igrnd_i$, the permittivity ϵ_r and conductivity σ are determined as follows:

For salt water ($igrnd_i = 0$), the relative permittivity is given by 70 for $f_{\text{MHz}} \leq 2253.5895$; and the conductivity is given by 5.0 S/m for $f_{\text{MHz}} \leq 1106.207$. For $f_{\text{MHz}} > 2253.5895$, the relative permittivity is given by

$$\epsilon_r = \left[\frac{1.4114535 \times 10^{-2} - 5.2122497 \times 10^{-8} f_{\text{MHz}} + 5.8547829 \times 10^{-11} f_{\text{MHz}}^2}{-7.6717423 \times 10^{-16} f_{\text{MHz}}^3 + 2.9856318 \times 10^{-21} f_{\text{MHz}}^4} \right]^{-1}.$$

For $f_{\text{MHz}} > 1106.207$, the conductivity σ in S/m is given by

$$\sigma = \frac{3.8586749 + 9.1253873 \times 10^{-4} f_{\text{MHz}} + 1.530992 \times 10^{-8} f_{\text{MHz}}^2}{1. - 2.1179295 \times 10^{-5} f_{\text{MHz}} + 6.5727504 \times 10^{-10} f_{\text{MHz}}^2 - 1.9647664 \times 10^{-15} f_{\text{MHz}}^3}.$$

For fresh water ($igrnd_i = 1$), the relative permittivity ϵ_r is given by 80 for $f_{\text{MHz}} \leq 6165.776$. For higher frequencies, ϵ_r is given by

$$\epsilon_r = \frac{79.027635 - 3.5486605 \times 10^{-4} f_{\text{MHz}} + 8.210184 \times 10^{-9} f_{\text{MHz}}^2}{1. - 2.2083308 \times 10^{-5} f_{\text{MHz}} + 2.7067836 \times 10^{-9} f_{\text{MHz}}^2 - 1.0007669 \times 10^{-14} f_{\text{MHz}}^3}.$$

For $f_{\text{MHz}} > 5776.157$, the conductivity σ in S/m is given by

$$\sigma = \left(\frac{-0.65750351 + 6.6113198 \times 10^{-4} f_{\text{MHz}} + 1.4876952 \times 10^{-9} f_{\text{MHz}}^2}{1. + 5.5620223 \times 10^{-5} f_{\text{MHz}} + 3.0140816 \times 10^{-10} f_{\text{MHz}}^2} \right)^2.$$

For $f_{\text{MHz}} \leq 5776.157$, the conductivity σ in S/m is given by

$$\sigma = \left(\frac{201.97103 + 1.2197967 \times 10^{-2} f_{\text{MHz}} - 1.728776 \times 10^{-6} f_{\text{MHz}}^2}{1. - 2.5539582 \times 10^{-3} f_{\text{MHz}} - 3.7853169 \times 10^{-5} f_{\text{MHz}}^2} \right)^{-1}.$$

For wet ground ($igrnd_i = 2$), the relative permittivity ϵ_r is given by 30 for $f_{MHz} \leq 1312.054$. For $1312.054 < f_{MHz} < 4228.11$, the relative permittivity ϵ_r is given by

$$\epsilon_r = \sqrt{\frac{857.94335 + 5.5275278 \times 10^{-2} f_{MHz}}{1. - 8.9983662 \times 10^{-5} f_{MHz} + 8.8247139 \times 10^{-8} f_{MHz}^2}} .$$

For $f_{MHz} \geq 4228.11$, the relative permittivity ϵ_r is given by

$$\epsilon_r = \sqrt{\frac{915.31026 - 4.0348211 \times 10^{-3} f_{MHz} + 7.4342897 \times 10^{-7} f_{MHz}^2}{1. - 9.4530022 \times 10^{-6} f_{MHz} + 4.892281 \times 10^{-8} f_{MHz}^2}} .$$

For $f_{MHz} > 15454.4$, the conductivity σ in S/m for wet ground is given by

$$\begin{aligned} \sigma = & 0.8756665 + 4.7236085 \times 10^{-5} f_{MHz} + 2.6051966 \times 10^{-8} f_{MHz}^2 \\ & - 9.235936 \times 10^{-13} f_{MHz}^3 + 1.4560078 \times 10^{-17} f_{MHz}^4 \\ & - 1.1129348 \times 10^{-22} f_{MHz}^5 + 3.3253339 \times 10^{-28} f_{MHz}^6 . \end{aligned}$$

For $f_{MHz} \leq 15454.4$, the conductivity σ in S/m for wet ground is given by

$$\begin{aligned} \sigma = & 5.5990969 \times 10^{-3} + 8.7798277 \times 10^{-5} f_{MHz} + 6.2451017 \times 10^{-8} f_{MHz}^2 \\ & - 7.1317207 \times 10^{-12} f_{MHz}^3 + 4.2515914 \times 10^{-16} f_{MHz}^4 \\ & - 1.240806 \times 10^{-20} f_{MHz}^5 + 1.3854354 \times 10^{-25} f_{MHz}^6 . \end{aligned}$$

For medium dry ground ($igrnd_i = 3$), the relative permittivity ϵ_r is given by 15 for $f_{MHz} \leq 4841.945$. For $f_{MHz} > 4841.945$, the relative permittivity ϵ_r is given by

$$\epsilon_r = \sqrt{\frac{215.87521 - 2.6151055 \times 10^{-3} f_{MHz} + 1.9484482 \times 10^{-7} f_{MHz}^2}{1. - 7.6649237 \times 10^{-5} f_{MHz} + 1.2565999 \times 10^{-8} f_{MHz}^2}} .$$

At $f_{MHz} \leq 4946.751$ for medium dry ground, the conductivity σ in S/m is given by

$$\begin{aligned} \sigma = & (2.4625032 \times 10^{-2} + 1.8254018 \times 10^{-4} f_{MHz} - 2.664754 \times 10^{-8} f_{MHz}^2 \\ & + 7.6508732 \times 10^{-12} f_{MHz}^3 - 7.4193268 \times 10^{-16} f_{MHz}^4)^2 . \end{aligned}$$

For $f_{MHz} > 4946.751$, for medium dry ground, the conductivity σ in S/m is given by

$$\sigma = (0.17381269 + 1.2655183 \times 10^{-4} f_{MHz} - 1.6790756 \times 10^{-9} f_{MHz}^2 + 1.1037608 \times 10^{-14} f_{MHz}^3 - 2.9223433 \times 10^{-20} f_{MHz}^4)^2.$$

For very dry ground ($igrnd_i = 4$), the relative permittivity ϵ_r is given by 3 and the conductivity σ in S/m is 0.0001 for $f_{MHz} < 590.8924$. For $590.8924 \leq f_{MHz} \leq 7131.933$, the conductivity σ in S/m is given by

$$\begin{aligned} \sigma = & 2.2953743 \times 10^{-4} - 8.1212741 \times 10^{-7} f_{MHz} + 1.8045461 \times 10^{-9} f_{MHz}^2 \\ & - 1.960677 \times 10^{-12} f_{MHz}^3 + 1.256959 \times 10^{-15} f_{MHz}^4 - 4.46811 \times 10^{-19} f_{MHz}^5 \\ & + 9.4623158 \times 10^{-23} f_{MHz}^6 - 1.1787443 \times 10^{-26} f_{MHz}^7 + 7.9254217 \times 10^{-31} f_{MHz}^8 \\ & - 2.2088286 \times 10^{-35} f_{MHz}^9. \end{aligned}$$

For $f_{MHz} > 7131.933$ MHz, the conductivity σ in S/m is given by

$$\sigma = (-4.9560275 \times 10^{-2} + 2.9876572 \times 10^{-5} f_{MHz} - 3.0561848 \times 10^{-10} f_{MHz}^2 + 1.1131828 \times 10^{-15} f_{MHz}^3)^2.$$

For ice at -1°C ($igrnd_i = 5$), the relative permittivity ϵ_r is 3 for all frequencies, and the conductivity σ , for $f_{MHz} \leq 300$, is given by

$$\sigma = \frac{3.8814567 \times 10^{-5} + 9.878241 \times 10^{-6} f_{MHz} + 7.9902484 \times 10^{-8} f_{MHz}^2}{1 + 8.467523 \times 10^{-2} f_{MHz} - 9.736703 \times 10^{-5} f_{MHz}^2 + 3.269059 \times 10^{-7} f_{MHz}^3},$$

and for $f_{MHz} > 300$ is given by

$$\sigma = \frac{1.2434792 \times 10^{-4} + 8.680839 \times 10^{-7} f_{MHz} + 7.2701689 \times 10^{-11} f_{MHz}^2 - 2.6416983 \times 10^{-14} f_{MHz}^3 + 1.37552 \times 10^{-18} f_{MHz}^4}{1 + 2.824598 \times 10^{-4} f_{MHz} - 6.755389 \times 10^{-8} f_{MHz}^2 + 2.8728975 \times 10^{-12} f_{MHz}^3 - 1.8795958 \times 10^{-18} f_{MHz}^4}.$$

For ice at -10°C ($igrnd_i = 6$), the relative permittivity ϵ_r is 3 for all frequencies, and the conductivity σ , for $f_{MHz} \leq 8753.398$, is given by

$$\sigma = \frac{1 + 3.883854 \times 10^{-2} f_{MHz} + 6.832108 \times 10^{-5} f_{MHz}^2}{51852.543 + 389.58894 f_{MHz}},$$

and for $f_{MHz} > 8753.398$, is given by

$$\sigma = 4.13105 \times 10^{-5} + 2.03589 \times 10^{-7} f_{MHz} - 3.1739 \times 10^{-12} f_{MHz}^2 + 4.52331 \times 10^{-17} f_{MHz}^3.$$

For the user-defined ground type ($igrnd_i = 7$), the relative permittivity ϵ_r and the conductivity σ in S/m are set equal to the input values $dielec_{1,i}$ and $dielec_{2,i}$, respectively.

Finally, the complex dielectric constant is given by

$$nc_i^2 = \epsilon_{ri} + 60\lambda\sigma_i; \text{ for } i = 1, 2, 3, \dots, i_{gr}.$$

Table 22 and Table 23 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the DIEINIT SU.

Table 22. DIEINIT SU input data element requirements.

Name	Description	Units	Source
<i>dielec</i>	2-dimensional array containing the relative permittivity and conductivity; $dielec_{1,i}$ and $dielec_{2,i}$, respectively.	N/A, S/m	Calling CSCI, DIEINIT SU
f_{MHz}	Frequency	MHz	APM_MOD
i_{gr}	Number of different ground types specified	N/A	Calling CSCI
<i>igrnd</i>	Integer array containing ground type composition for given terrain profile - can vary with range. Different ground types are: 0 = sea water 1 = fresh water 2 = wet ground 3 = medium dry ground 4 = very dry ground 5 = ice at -1 degree C 6 = ice at -10 degree C 7 = user defined (in which case, values of relative permittivity and conductivity must be given).	N/A	Calling CSCI
λ	Wavelength	meters	APMINIT CSC
<i>rgrnd</i>	Array containing ranges at which varying ground types apply.	meters	Calling CSCI

Table 23. DIEINIT SU output data element requirements.

Name	Description	Units
nc^2	Array of complex dielectric constants	N/A

5.1.8 FFT Parameters (FFTPAR) SU

The purpose of the FFTPAR SU is to determine the required transform size based on the maximum PE propagation angle and the maximum height needed for desired

coverage. If running in full or partial hybrid modes, the maximum height is the height necessary to encompass at least 20% above the maximum terrain peak along the path or the highest trapping layer specified in the environment profiles, whichever is greater. If running in a PE-only mode, the maximum height is the specified maximum output height.

For computational efficiency reasons, an artificial upper boundary is established for the PE solution. To prevent upward propagating energy from being “reflected” downward from this boundary and contaminating the PE solution, the PE solution field strength is attenuated or “filtered” above a certain height to ensure that the field strength just above this boundary is reduced to zero. The bin width in z-space Δz_{PE} is found from

$$\Delta z_{PE} = \frac{\lambda}{2 \sin(\theta_{max})}$$

where λ is the wavelength in meters and θ_{max} is the maximum propagation angle in radians.

The flag, i_{flag} , is used to determine maximum FFT size based on a given θ_{max} and desired coverage height z_{lim} or it will determine z_{lim} based on a given θ_{max} and FFT size.

For $i_{flag}=0$, the constants ln_{fft} , n_{fft} , and z_{max} are found from ln_{min} as follows,

$$\begin{aligned} ln_{fft} &= ln_{min}, \\ n_{fft} &= 2^{ln_{fft}}, \\ z_{max} &= n_{fft} \Delta z_{PE}, \end{aligned}$$

where ln_{min} is the minimum power of two transform size. ln_{min} is initialized to 10 and for every 5° in θ_{max} is increased by 1. Next, the transform size needed to perform calculations to a test height z_t is determined. First, z_t is set equal to z_{lim} minus a small height precision tolerance. Then a DO WHILE loop is executed as long as the condition $3/4 z_{max} < z_t$ is satisfied. Within this DO WHILE loop z_{max} is found from

$$\begin{aligned} ln_{fft} &= ln_{fft} + 1, \\ n_{fft} &= 2^{ln_{fft}}, \\ z_{maz} &= n_{fft} \Delta z_{PE}. \end{aligned}$$

If ln_{fft} reaches the value of 30, then the SU is exited with a non-zero error code.

For the case where $i_{flag}=1$, no iteration needs to be performed. z_{lim} is determined by a given ln_{fft} and θ_{max} from equations shown above.

Upon exiting, z_{lim} is computed as $\frac{3}{4} z_{max}$.

Table 24 and Table 25 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the FFTPAR SU.

Table 24. FFTPAR SU input data element requirements.

Name	Description	Units	Source
i_{flag}	Flag indicating whether to determine maximum FFT size n_{fft} based on given θ_{max} and z_{lim} or determine z_{lim} based on given θ_{max} and FFT size n_{fft} .	N/A	APMINIT CSC GETTHMAX SU
λ	Wavelength	meters	APMINIT SU
ln_{min}	Minimum power of 2 transform size	N/A	APMINIT SU
θ_{max}	Maximum propagation angle in PE calculations	radians	APMINIT CSC GETTHMAX SU
z_{lim}	Maximum height region where PE solution is valid	meters	APMINIT CSC GETTHMAX SU

Table 25. FFTPAR SU output data element requirements.

Name	Description	Units
Δz_{PE}	Bin width in z space	meters
i_{err}	Error code	N/A
ln_{fft}	Power of 2 transform size, i.e., $n_{fft}=2^{ln_{fft}}$	N/A
n_{fft}	Transform size	N/A
z_{lim}	Maximum height region where PE solution is valid	meters
z_{max}	Total height of the FFT/PE calculation domain	meters

5.1.9 Fill Height Arrays (FILLHT) SU

The purpose of the FILLHT SU is to calculate the effective earth radius for an initial launch angle of 5° and to fill an array with height values at each output range of the limiting sub-model, depending on which mode is being used. If running in a full hybrid mode, the array contains height values at each output range separating the PE from the RO region. If running in partial hybrid or PE-only modes, the array contains those height values at each output range at which the initial launch angle has been traced to the ground or surface. These height values represent the separating region where, above that height, valid loss is computed, and below that height, no loss is computed. This is done so that only loss values that fall within a valid calculation region are output.

Upon entering the SU, internal one-line ray trace functions are defined as

$$\begin{aligned}
\mathbf{RADA1}(a,b) &= a^2 + 2g_{rd}b, \\
\mathbf{RP}(a,b) &= a + \frac{b}{g_{rd}}, \\
\mathbf{AP}(a,b) &= a + bg_{rd}, \\
\mathbf{HP}(a,b) &= a + \frac{b^2 - c^2}{2g_{rd}},
\end{aligned}$$

for general parameters a , b , c , and refractivity gradient g_{rd} .

For the case when $i_{hybrid}=1$ (full hybrid mode), all height values at each output range separating the FE region from the RO region, are determined and stored in array $htfe$. For ranges greater than 2.5 km, the ray defined by a 5° elevation angle is traced up to the maximum height ht_{lim} . The ray is “traced” by simple geometry at every output range and the height array $htfe$ is determined as follows. The temporary variable y_{ar} is found from

$$y_{ar} = y_{fref} - ant_{ref},$$

where the parameter y_{fref} is the ground elevation height at the source, and ant_{ref} is the transmitting antenna height relative to the height h_{minter} . The values of $htfe_i$ are then determined by

$$\begin{aligned}
htfe_i &= y_{fref}; \quad \text{for } rngout_i \leq r_{tst} \\
htfe_i &= \mathbf{MIN}(ht_{lim}, \mathbf{MAX}\{y_{fref}, y_{ar} + t_5 rngout_i\}); \quad \text{for } rngout_i > r_{tst}, \quad i = 1, 2, \dots, n_{rout}
\end{aligned}$$

where r_{tst} is a constant range of 2,500 meters, t_5 is the tangent of 5° , and $rngout_i$ is the output range at every i^{th} range step.

For the airborne hybrid model ($i_{hybrid}=0$) the TRACE_ROUT SU is referenced to determine the heights at every output range separating the upper FE region from the PE region. These heights are stored in array $hlim$. The heights separating the lower FE region from the PE region are stored in array $htfe$ as outlined below.

For partial hybrid (PE plus XO) or airborne modes, the initial launch angle is traced until it hits the surface, storing heights traced at each output range.

First, several variables are initialized. The angle at the start of the trace, a_0 , is set to $-a_{launch}$ (determined in the GETTHMAX SU); the initial range, r_0 , is set equal to zero; and the height at the start of the ray trace step, h_0 , is set equal to ant_{ref} . The index, l ,

indicating the location of the source height in array $htdum$, is set equal to the index i_{startl} . The terrain elevation at the current range, ty_r , is initialized to 0, and the index j is set equal to one.

The following steps (1 through 3) are performed until the ray has reached the surface or the ray has been traced to r_{max} , whichever comes first.

1. The output range to trace to, r_o , is initialized to $rngout_j$, and $htfe_j$ is initialized to 0.
2. Now, the following steps 2a through 2d are performed until r_0 has reached r_o , the ray has turned around within a refractive layer, or the ray has hit the surface, whichever comes first.
 - a. The range at the end of the ray trace step, r_1 , is initialized to r_o . The refractivity gradient, g_{rd} is set equal to $grdum_{l-1}$. The angle and height at the end of the trace step, a_1 and h_1 , are now given by

$$\begin{aligned} a_1 &= AP(a_0, r_1 - r_0), \\ h_1 &= HP(h_0, a_1, a_0). \end{aligned}$$

- b. If a terrain profile has been specified ($f_{ter} = \text{'true.'}$), then the terrain elevation at the current traced range r_1 is determined according to

$$tyh_r = tyh_k + tyh_{k+1} \left(\frac{r_1 - r_k}{r_{k+1} - r_k} \right)$$

where tyh is the array of terrain elevations sampled at every PE range step Δr_{PE} and tyh_k is the terrain height at the k^{th} range step r_k equal to $k\Delta r_{PE}$.

- c. For negative values of a_1 , the height h_1 is first checked to determine if the ray height h_1 has fallen below the current terrain height tyh_r . If so, h_1 is set equal to tyh_r and a new angle and range are re-computed

$$\begin{aligned} a_1 &= -\sqrt{\mathbf{RADA1}(a_0, h_1 - h_0)}, \\ r_1 &= \mathbf{RP}(r_0, a_1 - a_0). \end{aligned}$$

If the ray has been traced through a lower refractive layer, then the index l is decremented by one and h_1 is now set to $htdum_l$. A new angle and range a_1 and r_1 are re-computed as above.

- d. a_0 , r_0 , and h_0 are now set equal to the values of a_1 , r_1 , and h_1 , respectively. If one of the conditions in step 2 has been met, then the SU proceeds to step 3; otherwise, steps 2a through 2d are repeated.
3. The ray has now been traced to output range $rngout_j$ and height h_0 at that range is stored in array $htfe$. The index j is incremented by one and if the ray has not reached the surface or r_{max} , then steps 1 through 3 are repeated.

Once the ray trace is completed, the index j is decremented by one and $htfe_j$ is set equal to hm_{ref} for all remaining output range steps j through n_{rout} .

Table 26 and Table 27 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the FILLHT SU.

Table 26. FILLHT SU input data element requirements.

Name	Description	Units	Source
a_{launch}	Launch angle used which, when traced, separates the PE and XO regions from the RO region	radians	GETTHMAX SU
ant_{ref}	Transmitting antenna height relative to the reference height h_{minter}	meters	TERINIT SU
Δr_{PE}	PE range step	meters	PEINIT SU
f_{ter}	Logical flag indicating if terrain profile has been specified: '.true.' = terrain profile specified '.false.' = terrain profile not specified	N/A	TERINIT SU
$grdum$	M-unit gradient array	(M-unit/meter)	REFINIT SU
$hmref$	Height relative to h_{minter}	meters	TERINIT SU
$htdum$	Height array for current interpolated profile	meters	REFINIT SU
ht_{lim}	User-supplied maximum height relative to h_{minter} , i.e., $ht_{lim} = h_{max} - h_{minter}$	meters	TERINIT SU
i_{hybrid}	Integer indicating which sub-models will be used: 0 = pure PE model 1 = full hybrid model (PE + FE + RO + XO) 2 = partial hybrid model (PE + XO)	N/A	APMINIT CSC
i_{start1}	Refractivity level index within $htdum$ at ant_{ref}	N/A	REFINIT SU
$lvlep$	Number of refractivity levels in profile $htdum$, $refdum$	N/A	REFINIT SU
n_{rout}	Integer number of the output range points desired	N/A	Calling CSCI
$rngout$	Array containing all output ranges	meters	APMINIT CSC

Table 26. FILLHT SU input data element requirements. (Continued)

Name	Description	Units	Source
r_{tst}	Range set at 2.5 km to begin calculation of RO values	meters	APM_MOD
θ_{75}	75% of maximum propagation angle in PE calculations	radians	APMINIT CSC
tyh	Adjusted height points of sampled terrain profile at every PE range step	meters	TERINIT SU
y_{fref}	Ground elevation height at the source	meters	APMINIT CSC

Table 27. FILLHT SU output data element requirements.

Name	Description	Units
$hlim$	Array containing height at each output range separating the RO region from the PE (at close ranges) and XO (at far ranges) regions	meters
$htfe$	Array of height values at each output range separating the PE region from the RO region	meters
i_{hmx}	Output range step index where height ht_{lim} is reached in array $hlim$	N/A

5.1.10 Gaseous Absorption (GASABS) SU

The purpose of the GASABS SU is to compute the specific attenuation based on air temperature and absolute humidity. This SU is based on CCIR (International Telecommunication Union, International Radio Consultative Committee, now the ITU-R) Recommendation 676-1, “Attenuation by Atmospheric Gases in the Frequency Range 1-350 GHz.”

The oxygen absorption for 15°C air temperature is computed from

$$\gamma_o = 10^{-3} (t_1 + t_2 + 0.00719) \left(\frac{f_{MHz}}{1000} \right)^2,$$

where f_{MHz} is the frequency in MHz and the temporary variables t_1 and t_2 are given by

$$t_1 = \frac{6.09}{\left(\frac{f_{MHz}}{1000} \right)^2 + 0.227},$$

$$t_2 = \frac{4.81}{\left(\frac{f_{MHz}}{1000} - 57.0 \right)^2 + 1.50}.$$

A correction is made for the oxygen absorption for the actual air temperature, which is given by

$$\gamma_o = (1.0 + 0.01 \{t_{air} - 15.0\}) \gamma_o$$

where t_{air} is the surface air temperature in degrees C.

The water vapor absorption is computed from

$$\gamma_w = \frac{(0.05 + 0.0021 abs_{hum} + t_1 + t_2 + t_3) \left(\frac{f_{MHz}}{1000} \right)^2 abs_{hum}}{10000.0}$$

where the temporary variables t_1 , t_2 , and t_3 are given, respectively, by

$$t_1 = \frac{3.6}{\left(\frac{f_{MHz}}{1000} - 22.2 \right)^2 + 8.5} ,$$

$$t_2 = \frac{10.6}{\left(\frac{f_{MHz}}{1000} - 183.3 \right)^2 + 9.0} ,$$

and

$$t_3 = \frac{8.9}{\left(\frac{f_{MHz}}{1000} - 325.4 \right)^2 + 26.3} .$$

The total specific absorption for sea-level air in dB/km multiplied by a conversion factor to convert to dB/m is given by

$$gas_{att} = (\gamma_o + \gamma_w) 10^{-3} .$$

Table 28 and Table 29 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the GASABS SU.

Table 28. GASABS SU input data element requirements.

Name	Description	Units	Source
abs_{hum}	Absolute humidity near the surface	g/meter ³	Calling CSCI
f_{MHz}	Frequency	MHz	Calling CSCI
t_{air}	Air temperature near the surface	°C	Calling CSCI

Table 29. GASABS SU output data requirements.

Name	Description	Units
gas_{att}	Gaseous absorption	dB/m

5.1.11 Get Effective Earth Radius Factor (GET_K) SU

The purpose of the GET_K SU is to compute the effective earth radius factor and the effective earth radius. The computation is made for a launch angle of 5° if the SU is called from the APMINIT CSC. If called from the TROPOINT SU, then the computation is made for a launch angle equal to the critical angle.

Upon entering the SU, internal one-line ray trace functions are defined as

$$\begin{aligned} \mathbf{RADA1}(a, b) &= a^2 + 2g_{rd}b, \\ \mathbf{RP}(a, b) &= a + \frac{b}{g_{rd}}, \end{aligned}$$

for general parameters a , b , and refractivity gradient g_{rd} .

The starting launch angle a_{start} for tracing a ray to determine the effective earth radius is initialized to the critical angle a_{crit} .

If the SU is called from the APMINIT CSC ($i_{org} = 0$), then a_{start} is re-initialized to 5°. If running the airborne model, then the beamwidth and antenna elevation angle are taken into account and the starting angle is initialized according to

$$a_{start} = \mathbf{MIN}[\mathbf{MAX}(a_{start}, \mu_{bwr} + \mu_{lim}), 10^\circ],$$

where

$$\mu_{lim} = \mathbf{MIN}(10^\circ, |\mu_{or}|).$$

If a terrain profile has been specified ($f_{ter} = \text{'true.'}$), then a_{start} is set equal to

$$a_{start} = \text{MIN}(1.5a_{start}, 10^\circ).$$

If using the airborne or full hybrid modes, or if calling from the TROPOINT SU, then the following steps 1 through 4 are performed to compute the effective earth radius from the antenna height up to height ht_{lim} .

1. The propagation angle, range, and height at the start of the ray trace step are initialized to a_{start} , 0., and ant_{ref} , respectively. The current refractivity level i is also initialized to i_{start1} .
2. The following steps 2a through 2c are performed for an upward ray until it has reached the last height in the refractivity level or ht_{lim} , whichever comes first.
 - a. The gradient g_{rd} is set equal to $grdum_i$. The propagation angle a_1 and range r_1 at the end of the trace step are computed as

$$a_1 = \sqrt{\text{RADA1}(a_0, htdum_{i+1} - h_0)},$$

$$r_1 = \text{RP}(r_0, a_1 - a_0).$$

- b. a_0 , r_0 , and h_0 are now set equal to the values of a_1 , r_1 , and $htdum_{i+1}$, respectively. If h_0 is greater than ht_{lim} , then the integer flag i_{flag} is set equal to 1, and the propagation angle a_1 at ht_{lim} is computed:

$$a_1 = \sqrt{\text{RADA1}(a_0, ht_{lim} - htdum_i)}.$$

A temporary maximum propagation angle Θ_{75a} is then set equal to a_1 .

- c. The current refractivity level i is incremented by 1. If one of the conditions in step 2 has been met, then the SU proceeds to step 3; otherwise, steps 2a through 2c are repeated.
3. If h_0 is less than ht_{lim} and i_{flag} is equal to 0, then propagation angle a_1 at ht_{lim} is re-computed as

$$a_1 = \sqrt{\text{RADA1}(a_0, ht_{lim} - h_0)},$$

and the variable Θ_{75a} is then set equal to a_1 .

4. The propagation angle and range a_1 and r_1 are now re-computed from

$$a_1 = \sqrt{\mathbf{RADA1}(a_0, htdum_{lvlep} - h_0)},$$

$$r_1 = \mathbf{RP}(r_0, a_1 - a_0),$$

and the effective earth radius a_{ek} , the effective earth radius factor e_k , and twice the effective earth radius, $twoka$ are given by

$$a_{ek} = \frac{r_1}{a_1 - a_{start}},$$

$$twoka = 2 a_{ek},$$

$$e_k = 6.37 \times 10^{-6} a_{ek}.$$

If using the airborne hybrid model and the calling SU is the APMINIT CSC, then twice the effective earth radius factor is computed for a downward ray where the initial launch angle is $-a_{start}$. Steps 1 through 2a are repeated with a_1 negative, i decremented by 1, and $htdum_{i+1}$ replaced with $htdum_i$. Finally, the variable $twoka_{down}$ is computed from

$$twoka_{down} = \frac{2 r_1}{a_1 + a_{start}},$$

and Θ_{75} is determined from

$$\Theta_{75} = \mathbf{MAX}(\Theta_{75a}, a_{start}).$$

Table 30 and Table 31 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the GET_K SU.

Table 30. GET_K SU input data element requirements.

Name	Description	Units	Source
a_{crit}	Critical angle (angle above which no rays are trapped)	radians	REFINIT SU
a_{launch}	Launch angle used which, when traced, separates the PE and XO regions from the RO region	radians	GETTHMAX SU
ant_{ref}	Transmitting antenna height relative to the reference height h_{minter}	meters	TERINIT SU
μ_{or}	Antenna pattern elevation angle	radians	APMINIT CSC
f_{ter}	Logical flag indicating if terrain profile has been specified: '.true.' = terrain profile specified '.false.' = terrain profile not specified	N/A	TERINIT SU
$grdum$	M-unit gradient array	(M-unit/ meter)	REFINIT SU
$hmref$	Height relative to h_{minter}	meters	TERINIT SU
$htdum$	Height array for current interpolated profile	meters	REFINIT SU
ht_{lim}	User-supplied maximum height relative to h_{minter} , i.e., $ht_{lim} = h_{max} - h_{minter}$	meters	TERINIT SU
i_{hybrid}	Integer indicating which sub-models will be used: 0 = pure PE model 1 = full hybrid model (PE + FE + RO + XO) 2 = partial hybrid model (PE + XO)	N/A	APMINIT CSC
i_{org}	Integer flag indicating origin of calling SU 0 = called from APMINIT CSC 1 = called from TROPOINT SU	N/A	APMINIT CSC TROPOINT SU
i_{start1}	Refractivity level index within $htdum$ at ant_{ref}	N/A	REFINIT SU
$lvlep$	Number of refractivity levels in profile $htdum$, $refdum$	N/A	REFINIT SU
y_{tref}	Ground elevation height at the source	meters	APMINIT CSC

Table 31. GET_K SU output data element requirements.

Name	Description	Units
a_{ek}	Effective earth radius	meters
e_k	Effective earth radius factor	N/A
θ_{75}	75% of maximum propagation angle in PE calculations	radians
$twoka$	Twice the effective earth radius	meters
$twoka_{down}$	Twice the effective earth radius for downward path	meters

5.1.12 Get Alpha Impedance (GETALN) SU

The purpose of the GETALN SU is to compute the surface impedance term in the Leontovich boundary condition and the complex index of refraction for finite conductivity. The implementation of these impedance formulas follow Kuttler and Dockery's method (Ref.8).

Upon entering the SU, the smooth surface impedance term, α , is computed from the complex dielectric constant nc^2 and free-space wave number k_o , for both vertical and horizontal polarization, by

$$\alpha_h = ik_o \sqrt{nc_{ig}^2 - 1},$$

$$\alpha_v = ik_o \frac{\sqrt{nc_{ig}^2 - 1}}{nc_{ig}^2}$$

where i is the imaginary number $\sqrt{-1}$.

If rough surface calculations are required ($ruf = \text{'true.'}$) and a non-zero grazing angle ψ exists for the current range step, then the rough surface reflection coefficient, Γ , is determined from referencing the GETREFCOEF SU and the surface impedance is re-computed as

$$\alpha_{h,v} = ik_o \text{SIN } \psi \frac{1 - \Gamma_{h,v}}{1 + \Gamma_{h,v}},$$

where the subscripts h,v indicate horizontal and vertical polarization quantities, respectively.

If using the central difference algorithm ($i_{alg} = 1$), the following steps 1 through 2 are performed to compute constants and variables for subsequent use in the MIXEDFT SU.

1. The determination of the complex root, R_T , of the quadratic equation for the mixed transform method is based on Kuttler's formulation

$$R_T = -\sqrt{1.0 + (\alpha_h \Delta z_{PE})^2} - \alpha_h \Delta z_{PE} \text{ for horizontal polarization,}$$

$$R_T = \sqrt{1.0 + (\alpha_v \Delta z_{PE})^2} - \alpha_v \Delta z_{PE} \text{ for vertical polarization}$$

2. Next, the array rn is determined according to

$$rn_i = R_T^i \quad i = 1, 2, \dots, n_{fft}.$$

Several parameters used in the central difference algorithm are now computed:

$$R_k = \frac{2(1 - rn_2)}{(1 + rn_2)(1 - rn_{n_{fft}}^2)},$$

$$C_{lx} = e^{i\Delta r_{PE} \sqrt{k_o^2 + \left(\frac{\text{LN}(R_T)}{\Delta z_{PE}}\right)^2}},$$

$$C_{2x} = e^{i\Delta r_{PE} \sqrt{k_o^2 + \left(\frac{\text{LN}(R_T) - i\pi}{\Delta z_{PE}}\right)^2}}.$$

If using the backward difference algorithm ($i_{alg} = 2$) R_T is computed as

$$R_T = (1 + \alpha_{h,v} \Delta z_{PE}),$$

the array rn is computed as in step 2 above and the parameter $cmft_x$ is computed using the same equation for C_{lx} in step 2 above.

Table 32 and Table 33 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the GETALN SU.

Table 32. GETALN SU input data element requirements.

Name	Description	Units	Source
Δr_{PE}	PE range step	meters	APMINIT CSC
Δz_{PE}	Bin width in z space	meters	FFTPAR SU
i_{alg}	Integer flag indicating which DMFT algorithm is being used: 0 = no DMFT algorithm will be used 1 = use central difference algorithm 2 = use backward difference algorithm	N/A	APMINIT CSC
i_g	Counter indicating current ground type being modeled	N/A	APMINIT CSC PESTEP SU
i_{pol}	Polarization flag: 0 = horizontal polarization 1 = vertical polarization	N/A	Calling CSCI
k_o	Free-space wave number	meters ⁻¹	APMINIT CSC

Table 32. GETALN SU input data element requirements. (Continued)

Name	Description	Units	Source
nc^2	Array of complex dielectric constants	N/A	DIEINIT SU
n_{fft}	Transform size	N/A	FFTPAR SU
ψ	Grazing angle	radians	Calling SU
r	Current calculation range	meters	Calling SU
ruf	Logical flag indicating if rough sea surface calculations are required 'true.' = perform rough sea surface calculations 'false.' = do not perform rough sea surface calculations	N/A	APMINIT CSC

Table 33. GETALN SU output data element requirements.

Name	Description	Units
$\alpha_{h,v}$	Surface impedance term for horizontal and vertical polarization	N/A
C_{1x}	Constant used to propagate c_{k1} by one range step in central difference algorithm	N/A
C_{2x}	Constant used to propagate c_{k2} by one range step in central difference algorithm	N/A
$cmft_x$	Constant used to propagate $cmft$ by one range step in backward difference algorithm	N/A
R_k	Coefficient used in c_{k1} and c_{k2} calculations.	N/A
m	Array of R_T to the i^{th} power (e.g., $m_i = R_T^i$)	N/A
R_T	Complex root of quadratic equation for mixed transform method based on Kuttler's formulation	N/A

5.1.13 Get Grazing Angle (GETGRAZE) SU

The purpose of the GETGRAZE SU is to compute grazing angles at each PE range step via ray trace and spectral estimation for subsequent use in rough sea surface calculations. This SU is referenced only if rough surface calculations are required ($ruf = \text{'true.'}$)

Upon entering the SU, the RDTRACE SU is referenced to determine the grazing angles ψ_{ray} from ray trace.

If a terrain profile has been specified ($f_{ter} = \text{'true.'}$) and a surface-based duct has been specified ($l_{duct} = \text{'true.'}$), then the grazing angles ψ_{PE} are computed from spectral estimation of the near-surface PE field by running the PE algorithm out to the maximum range r_{max} , assuming horizontal polarization and smooth surface conditions (i.e., no wind speed).

First, the array ψ_{PE} is allocated for size of i_{PE} – equal to the number of PE range steps required to propagate the field out to r_{max} . The array is initialized to zero, with the first element initialized to $\pi/2$ radians. The current PE range r and PE integer step i_{PEstp} are then set equal to zero, with the terrain height y_{last} at the previous range step set equal to tyh_0 if a terrain profile has been specified, or 0 otherwise. An iterative DO WHILE loop is then begun to advance the PE solution such that for the current PE range, a PE solution is calculated from the solution at the previous PE range. This iterative procedure is repeated in the DO WHILE loop until r is greater than r_{max} . The following steps (1 through 5) are performed for each PE range step within the DO WHILE loop.

1. The current PE calculation range r is incremented by one PE range step, Δr_{PE} , and the PE range step counter i_{PEstp} is incremented by 1. The range at which interpolation for range-dependent refractivity profiles is performed, r_{mid} , is also incremented by one-half the PE range step.
2. If performing a terrain case ($f_{ter} = \text{'true.'}$), the ground heights, y_{cur} and y_{curm} , at range r and r_{mid} , respectively, are determined according to

$$\begin{aligned} y_{cur} &= tyh_{i_{PEstp}}, \\ y_{curm} &= \frac{1}{2} (tyh_{i_{PEstp}-1} + y_{cur}). \end{aligned}$$

If y_{cur} is less than y_{curm} , the DOSHIFT SU is referenced to adjust the PE field relative to the terrain height.

The PE field array U is now propagated in free space one range step by referencing the FRSTP SU.

If the APM CSCI is being used in a range-dependent mode (i.e., the number of profiles n_{prof} is greater than 1), or a terrain profile is specified, the REFINITER SU is referenced to compute a new modified refractive index profile, $profint$, adjusted by the local ground height y_{curm} at range r_{mid} . A new environmental phase array, $envpr$, based on this new refractivity profile is then computed from

$$envpr_j = e^{i \Delta r_{PE} profint_j}; \quad j=1,2,\dots,n_{fft}$$

$$envpr_j = filt_{j-n_{34}} envpr_j; \quad j= n_{34}, n_{34}+1, n_{34}+2, \dots, n_{fft}$$

3. The complex field U is now multiplied by the environmental phase array for all bins from 0 through $n_{fft}-1$.

4. Next, if a terrain profile has been specified and the terrain slop is positive ($y_{cur} > y_{curm}$), the DOSHIFT SU is referenced to adjust the PE field relative to the terrain height.
5. The SPECEST SU is then referenced to determine the grazing angle ϑ_{out} , and this angle is stored in array ψ_{PE} . If no terrain has been specified and the range is greater than the horizon range r_{hor} , then the grazing angle stored is the smaller of the tangent angle a_{cut} or ϑ_{out} .

Finally, i_{gPE} , the number of grazing angles computed, is initialized to i_{PE} and the SU is exited.

Table 34 and Table 35 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the GETGRAZE SU.

Table 34. GETGRAZE SU input data element requirements.

Name	Description	Units	Source
a_{cut}	Tangent angle from antenna height to radio horizon	radians	PEINIT SU
ant_{ref}	Transmitting antenna height relative to the reference height h_{minter}	meters	TERINIT SU
Δr_{PE}	PE range step	meters	PEINIT SU
Δr_{PE2}	$\frac{1}{2}$ PE range step	meters	PEINIT SU
$filt$	Cosine-tapered (Tukey) filter array	N/A	PEINIT SU
f_{ter}	Logical flag indicating if terrain profile has been specified: '.true.' = terrain profile specified '.false.' = terrain profile not specified	N/A	TERINIT SU
h_{trap}	Height of the highest trapping layer from all refractivity profiles	meters	REFINIT SU
i_{hybrid}	Integer indicating which sub-models will be used: 0 = pure PE model 1 = full hybrid model (PE + FE + RO + XO) 2 = partial hybrid model (PE + XO)	N/A	APMINIT CSC
i_{PE}	Number of PE range steps	N/A	PEINIT SU
l_{duct}	Logical flag indicating if surface-based duct profile has been specified '.true.' = surface-based duct exists '.false.' = no surface-based duct exists	N/A	REFINIT SU
l_{evap}	Logical flag indicating if evaporation duct profile has been specified '.true.' = evaporation duct exists '.false.' = no evaporation duct exists	N/A	REFINIT SU
n_{fft}	Transform size	N/A	FFTPAR SU
n_{34}	$\frac{3}{4} n_{fft}$	N/A	PEINIT SU

Table 34. GETGRAZE SU input data element requirements. (Continued)

Name	Description	Units	Source
n_{f4}	$\frac{1}{4} n_{ft}$	N/A	PEINIT SU
n_{prof}	Number of refractivity profiles	N/A	Calling CSCI
r_{hor}	Radio horizon range	meters	PEINIT SU
r_{max}	Maximum specified range	meters	Calling CSCI
ϑ_{mxg}	Maximum PE calculation angle for spectral estimation of grazing angles	radians	APMINIT CSC
tyh	Adjusted height points of sampled terrain profile at every PE range step	meters	TERINIT SU

Table 35. GETGRAZE SU input data element requirements.

Name	Description	Units
ψ_{PE}	Array containing grazing angles computed from spectral estimation of PE field	radians
ψ_{ray}	2-dimensional array containing grazing angles and corresponding ranges computed from ray trace	radians, meters
i_{error}	Integer variable indicating error number for ALLOCATE and DEALLOCATE statements	N/A
i_{gPE}	Number of grazing angles computed from spectral estimation	N/A
i_{grz}	Number of grazing angles computed from ray trace	N/A

5.1.14 Get Maximum Angle (GETTHMAX) SU

The purpose of the GETTHMAX SU is to perform an iterative ray trace to determine the minimum angle required (based on the reflected ray) in obtaining a PE solution. The determination of this angle depends on the particular mode of execution. For full and partial hybrid modes, a ray is traced up to a height that exceeds at least 20% above the maximum terrain peak along the path or the highest trapping layer specified in the environment profiles, whichever is greater. Heights and angles of this ray are stored at each output range. The maximum PE propagation angle, θ_{max} , is then determined from the local maximum angle of the traced ray. For the full hybrid mode, the minimum PE propagation angle is required to meet the following criteria: (1) the top of the PE region must contain all trapping layers for all refractivity profiles; (2) the top of the PE region must be at least 20% higher than the highest peak along the terrain profile; and (3) the minimum PE propagation angle must be at least as large as the grazing angle of the limiting ray ψ_{lim} .

First, four in-line ray trace functions are defined for general parameters a , b , c , and g_{rd} :

$$\text{RADA1}(a,b)=a^2+2g_{rd}b,$$

$$\begin{aligned}\mathbf{RP}(a,b) &= a + \frac{b}{g_{rd}}, \\ \mathbf{AP}(a,b) &= a + b g_{rd}, \\ \mathbf{HP}(a,b,c) &= a + \frac{b^2 - c^2}{2g_{rd}}.\end{aligned}$$

The first parameter to be determined is the minimum PE angle limit a_{mlim} . The parameter to be determined later, Θ_{max} , must be at least this value. The initial estimate of a_{mlim} is given by

$$a_{mlim} = \pi/90 \left(.37541 + 4.331e^{\frac{-f_{MHz}}{248.4}} + 1.42e^{\frac{-f_{MHz}}{2867}} + .4091e^{\frac{-f_{MHz}}{2495}} \right).$$

If the DMFT algorithm is required based on the specified inputs, the antenna height is greater than 5 meters, or the frequency is greater than 15 GHz, then a_{mlim} is decreased by $1/2$.

If the central difference DMFT algorithm will be used, then a_{mlim} is adjusted to accommodate low antenna heights according to

$$a_{mlim} = \mathbf{MAX} \left(a_{mlim}, \mathbf{SIN}^{-1} \left(\frac{\lambda}{2 ant_{ht}} \right) \right).$$

A multiplicative height factor h_{mt} is determined to ensure clearance of the ray path for low antenna heights over large terrain elevations:

$$h_{mt} = \mathbf{MIN} \left(.2, \frac{ant_{ht}}{\mathbf{MAX}(ty_1, 1)} \right).$$

If h_{mt} is less than .1, then it is set equal to 0. It is then increased by 1.

Several constants needed in subsequent steps in this SU are determined. An initial estimate of the launch angle a_{launch} , is initialized to α_{lim} , the elevation angle of the RO limiting ray. If using the full hybrid mode, then a_{launch} is set equal to the negative of a_{launch} . The maximum height to trace to, z_{lim} , is set equal to $ht_{lim} \cdot 10^{-5}$, and the range step, Δr_{temp} , for subsequent ray tracing is given by $r_{max}/200$. The terrain elevation height at the source, y_{nt} , is initialized to ty_1 provided APM is running in a full hybrid mode and ty_1 is greater than zero; otherwise, y_{nt} is initialized to 0.

An iterative ray trace to determine the launch angle a_{launch} and subsequently Θ_{max} is then begun. The following steps steps 1 through 3 are performed until a ray has been safely traced from height ant_{ref} to z_{lim} .

1. At the start of the ray trace, the current local angle (a_0), range (r_0), height (h_0), and refractive gradient index (j) are initialized to a_{launch} , 0, ant_{ref} , and i_{start1} , respectively. The counter index, kt , for the terrain profile arrays tx and ty is initialized to one. The variable r_o , the current output range to trace to, is set equal to zero. The following steps 1.a through 1.d are then performed for each ray trace step from 1 to i_{rtemp} .

- a. First, r_o is incremented by Δr_{temp} . Now the following steps i through vii are performed until r_0 reaches r_o .

- i. The range at the end of the ray trace step, r_1 is set equal to r_o , and the current refractive gradient g_{rd} is set equal to $grdum_j$

- ii. The angle at the end of the trace, a_1 , is then given by

$$a_1 = AP(a_0, r_1 - r_0) .$$

- iii. If a_1 is of the opposite sign of a_0 , then a_1 is set to zero, and r_1 is given by

$$r_1 = RP(r_0, a_1 - a_0) .$$

- iv. The height at the end of the ray trace h_1 is given by

$$h_1 = HP(h_0, a_1, a_0) .$$

- v. If a_1 is positive and h_1 has reached or surpassed the next height level, then a_1 , r_1 , j , and h_1 , are found as follows. First, h_1 is set equal to $htdum_{j+1}$, and a_1 and r_1 are given by

$$\begin{aligned} a_1 &= \sqrt{RADA1(a_0, h_1 - h_0)} \\ r_1 &= RP(r_0, a_1 - a_0) \end{aligned} .$$

The index j is incremented by one, and the height, h_1 , at the end of the ray trace step is given by the smaller of ht_{lim} or $htdum_j$.

- vi. However, if either of the conditions for a_1 and h_1 in step v are not met, and a_1 is less than or equal to 0, then h_1 is set equal to y_{nt} if the calculated value in

step iv is less than y_{nt} . If the calculated value of h_1 in step iv is less than $htdum_j$, then h_1 is set equal to $htdum_j$, and j is set equal to the maximum of 0 or $j-1$. The variables a_1 and r_1 are then determined from

$$\begin{aligned} a_1 &= -\sqrt{\mathbf{RADA1}(a_0, h_1 - h_0)} \\ r_1 &= \mathbf{RP}(r_0, a_1 - a_0) \end{aligned}$$

vii. If the ray has hit the surface and is reflected, which would be the condition for which h_1 is set equal to y_{nt} in step vi, then a_1 is set equal to minus a_1 , ψ_{lim} is set equal to a_1 , the range, r_{pest} (at which loss values from the PE model will start being calculated) is set equal to r_1 , and the height h_{start} is set equal to y_{nt} . The variable h_{start} is used for subsequent initialization of ray tracing to fill in array h_{lim} . In preparation for the next ray trace step, h_0 is set equal to h_1 , r_0 is set equal to r_1 , and a_0 is set equal to a_1 . If the range r_1 is greater than r_{flat} , then the current iteration is exited and the SU proceeds to step b; otherwise, steps i through vii are repeated until r_0 reaches r_0 .

- b. If running a terrain case ($f_{ter} = \text{'true.'}$), at the end of the ray trace for the current step a check is made to see that the current height of the ray is at least 20% higher than the current terrain height. The counter kt is determined such that $r_0 > tx_{kt+1}$ and $kt < i_{tpa}$. If using the partial hybrid mode and range r_0 is less than 5 km, then the clearance height of the terrain, y_n , at the current range for the traced ray, is given by

$$y_n = h_{mt} [ty_{kt} + slp_{kt} (r_0 - tx_{kt})].$$

If the previous conditions are not met, then h_{mt} in the above equation is replaced with the constant 1.2.

- c. The ending angle, range, and height for each ray trace step is now stored in arrays $raya$, $rtemp$, and $htemp$, respectively.
- d. Now, if running a full hybrid case ($i_{hybrid} = 1$), a test is made to determine if both h_0 is less than y_n and if r_0 is greater than r_{flat} . If these conditions are true, then the flag i_{quit} is set equal to 1. If the case is not a full hybrid case and if h_0 is less than y_n , then i_{quit} is set equal to 1. Finally, if h_0 is greater than or equal to z_{limit} ; or i_{quit} equals 1, then the current iteration is exited and the SU proceeds to step 2; otherwise, the steps 1.a through 1.d are repeated.
2. If the iteration defined by steps 1.a through 1.d has been prematurely terminated ($i_{quit}=1$), then the initial elevation angle a_{launch} is decreased by 10^{-3} radians for the full hybrid case ($i_{hybrid}=1$), and is increased by 10^{-3} , otherwise. If the previous iteration has not been prematurely terminated ($i_{quit}=0$), the SU continues with step 3.

3. If height z_{lim} is reached, then an initial launch angle (i.e., ray) has been found with all traced heights, ranges, and angles stored. The integer flag to continue ray tracing, i_{ray} , is set to equal 1 to terminate the iterative loop, and the index i_{hmax} , indicating the range step at which z_{lim} is reached, is set equal to the range step index i (the range step index counter in the iterative loop defined by steps 1 through 3).

The remaining elements from i_{hmax} to i_{rtemp} in arrays h_{temp} , r_{temp} , and $raya$ are filled with the values h_0 , r_{max} , and a_0 , respectively. Next, the index i_{hmax} is set equal to the minimum of i_{hmax} or i_{rtemp} .

The variable Θ_{max} is found for the PE region based on the local ray angles just determined for the particular ray traced. First, the index i_{ap} at which the local ray angle becomes positive (i.e., $raya_{i_{ap}}$) is determined. If i_{ap} equals i_{rtemp} this indicates that no PE calculations are required for the specific geometry, in which case the flag no_{PE} is set equal to 1, Θ_{75} is set equal to a_{mlim} , r_{pest} is set equal to r_{max} , z_{lim} is set equal to ht_{lim} , and the SU is exited. Otherwise, several variables are next initialized. The local indices, i_{ok} and i_{flag} , plus the variables z_{lim} and a_{mxcur} , are each set equal to zero. The variable a_{mxcur} is the maximum local angle along the traced ray up to height z_{lim} with a minimum limit of a_{mlim} .

The variable Θ_{max} is then found from an iteration performed on the local angle and height at which the local maximum angle is reached. The following steps 1 through 6 are performed while the flag i_{ok} is 0.

1. The height in the PE region that must be reached for the hybrid model is z_{test} . The first occurrence of h_{temp}_j that is greater than z_{test} is found and the index i_{st} is then set to the smaller of the index j where this occurs or i_{hmax} .
2. The angle a_{mxcur} is now initialized to $raya_1$. The maximum angle in $raya$ is then found looking only at elements from $raya_2$ to $raya_{i_{st}}$ and a_{mxcur} is set equal to this angle.
3. a_{mxcur} is now set equal to the maximum of a_{mlim} and a_{mxcur} . The variable a_{temp} is now set to a_{mxcur} divided by 0.75. If using the partial hybrid mode ($i_{hybrid}=2$), z_{test} is given by

$$z_{test} = \text{MAX}(ant_{ref}, h_{test}, 1.2 h_{termax}, 10^3).$$

4. A reference is then made to the FFTPAR SU to determine new values for z_{test} , z_{max} , Δz_{PE} , ln_{ffi} , and n_{ffi} using the inputs: ln_{min} , λ , a_{temp} , and i_{flag} .
5. After the reference to the FFTPAR SU is made, if $i_{flag} = 0$ it is set equal to 1. In addition, if not running a full hybrid case, i_{ok} is set equal to 1. However, if after the

reference to the FFTPAR SU is made, i_{flag} is equal to one, and if the case is not a partial hybrid case, the iterative height tolerance tol is given by

$$tol = \frac{|z_{test} - z_{lim}|}{z_{test}} .$$

A test is then made to determine whether this value of tol is less than or equal to z_{tol} , the height tolerance for Newton's method. If it is, then the index i_{ok} is set equal to one.

6. Now z_{lim} is set equal to z_{test} , and if i_{ok} is 0, steps 1 through 6 are repeated. Otherwise, the SU proceeds to the next step.

The variable Θ_{75} is now set equal to a_{mxcur} , and Θ_{max} is set equal to a_{temp} . The variable ln_{fft} is then adjusted such that for every 5° in Θ_{75} , it is increased by 1.

Next, the TRACE_ROUT SU is referenced to trace the ray to each output range step, Δr_{out} , storing heights in array $hlim$. If running in a full hybrid mode, the ray is traced from starting angle, range, and height equal to ψ_{lim} , r_{pest} , and h_{start} , respectively. Otherwise, the starting angle, range, and height are equal to a_{launch} , 0, and ant_{ref} , respectively.

Before exiting, all elements of $hlim$ corresponding to ranges less than r_{pest} are set equal to y_{fref} .

Table 36 and Table 37 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the GETTHMAX SU.

Table 36. GETTHMAX SU input data element requirements.

Name	Description	Units	Source
α_{lim}	Elevation angle of the RO limiting ray	radians	Calling SU
ant_{ht}	Transmitting antenna height above local ground	meters	Calling CSCI
ant_{ref}	Transmitting antenna height relative to h_{minter}	meters	TERINIT SU
f_{MHz}	Frequency	MHz	Calling CSCI
f_{ter}	Logical flag indicating if terrain profile has been specified: '.true.' = terrain profile specified '.false.' = terrain profile not specified	N/A	TERINIT SU
$grdum$	M-unit gradient array	(M-unit/ meter)	REFINIT SU
$htdum$	Height array for current interpolated profile	meters	REFINIT SU
h_{termax}	Maximum terrain height along profile path	meters	Calling SU
h_{test}	Minimum height at which all trapping refractivity features are below	meters	Calling SU
$htlim$	User-specified maximum height relative to h_{minter}	meters	TERINIT SU
$ihybrid$	Integer indicating which sub-models will be used: 0 = pure PE model 1 = full hybrid model (PE + FE + RO + XO) 2 = partial hybrid model (PE + XO)	N/A	APMINIT CSC
i_{alg}	Integer flag indicating which DMFT algorithm is being used: 0 = no DMFT algorithm will be used 1 = use central difference algorithm 2 = use backward difference algorithm	N/A	APMINIT CSC
i_{temp}	Temporary number of range steps (used for ray tracing)	N/A	APM_MOD
i_{start1}	Refractivity level index within $htdum$ at ant_{ref}	N/A	REFINIT SU
i_{tpa}	Number of terrain points in used internally in arrays tx and ty	N/A	APMINIT CSC
λ	Wavelength	meters	APMINIT CSC
$lnmin$	Minimum power of 2 transform size	N/A	APMINIT CSC
n_{rout}	Integer number of output range points desired	N/A	Calling CSCI
r_{flat}	Maximum range at which the terrain profile remains flat from the source	meters	Calling SU
r_{max}	Maximum output range	meters	Calling CSCI
$rngout$	Array containing all desired output ranges	meters	APMINIT CSC
ruf	Logical flag indicating if rough sea surface calculations are required '.true.' = perform rough sea surface calculations '.false.' = do not perform rough sea surface calculations	N/A	APMINIT CSC
slp	Slope of each segment of terrain	N/A	TERINIT SU
tx	Range points of terrain profile	meters	TERINIT SU

Table 36. GETTHMAX SU input data element requirements. (Continued)

Name	Description	Units	Source
ty	Adjusted height points of terrain profile	meters	TERINIT SU
y_{fref}	Ground elevation height at source	meters	APMINIT CSC
z_{lim}	Height limit for PE calculation region	meters	APMINIT CSC
z_{test}	Height in PE region that must be reached for hybrid model	meters	Calling SU
z_{tol}	Height tolerance for Newton's method	meters	APMINIT CSC

Table 37. GETTHMAX SU output data element requirements.

Name	Description	Units
a_{launch}	Launch angle used which, when traced, separates PE and XO regions from the RO region	radians
$hlim$	Array containing height at each output range separating the RO region from the PE (at close ranges) and XO (at far ranges) regions	meters
$htemp$	Heights at which ray is traced to every range in $rtemp$	meters
i_{ap}	Index indicating when the local ray angle becomes positive in array $raya$	N/A
i_{err}	Return error code	N/A
ln_{fft}	Power of 2 transform size, i.e., $n_{fft}=2^{ln_{fft}}$	N/A
no_{PE}	Integer flag indicating if PE calculations are needed: 0 = PE calculations needed 1 = no PE calculations needed	N/A
ψ_{lim}	Grazing angle of limiting ray	radians
$raya$	Array containing all local angles of traced ray a_{launch} at each i_{rtemp} range	radians
r_{pest}	Range at which loss values from the PE model will start being calculated	meters
$rtemp$	Range steps for tracing to determine maximum PE angle	meters
θ_{max}	Maximum propagation angle in PE calculations	radians
θ_{75}	75% of maximum propagation angle in PE calculations	radians
z_{max}	Maximum height in PE calculation region	meters

5.1.15 Grazing Angle Interpolation (GRAZE_INT) SU

The purpose of the GRAZE_INT SU is to interpolate for each PE range step grazing angles computed from both ray trace and spectral estimation. Those angles from ray trace take precedence.

Upon entering the SU, the grazing angle array Ψ is allocated for size i_{PE} and initialized to 0, with the first element initialized to $\frac{1}{2}\pi$ radians. Several variables are next initialized. The variable r which is the range to interpolate to, is initialized to Δr_{PE} . The

range r_{grz} at which the spectrally estimated angles were computed in the GETGRAZE SU is initialized to Δr_{grz} .

If a surface-based duct has been specified, no evaporation duct exists, and the range r_{flat} is greater than the horizon range r_{hor} , then a check is made for the possible existence of a skip zone produced by a surface-based duct. If one exists, then the spectrally estimated grazing angles will be included in the interpolation algorithm for those ranges beyond the start of the skip zone. The check for a skip zone is done by performing an iterative loop on the ranges r_{ray} corresponding to the grazing angles in ψ_{ray} . For those ranges beyond r_{hor} the maximum difference between successive ranges in r_{ray} is determined according to

$$r_{skip} = \text{maximum of } (r_{ray_{j+1}} - r_{ray_j}); \quad \text{for } j = k-1, k, \dots, i_{grz}-1,$$

where k is the first element in r_{ray} corresponding to the first range beyond r_{hor} . If r_{skip} is greater than 5 km, then a skip zone is assumed to exist and the range per at which spectrally estimated grazing will be included in the interpolation algorithm is set equal to the minimum of r_{flat} or r_{end} , where r_{end} is the first range point in r_{ray} just beyond the skip zone. If no skip zone exists, then per is set equal to r_{max} . Next, the following steps 1 through 3 are performed for each PE range step i , indexed from 1 to i_{PE} .

1. For range r less than r_{hor} steps 1.a through 1.c are performed; otherwise, the SU proceeds to step 2.
 - a. An iterative loop is performed to find k , the element in r_{ray} corresponding to the first range point beyond r_{hor} .
 - b. For k equal to 1 the grazing angle is determined as

$$\psi_i = \left| \text{TAN}^{-1} \left(\frac{s}{r} \right) \right|,$$

$$s = h_{mref} - y_{fref} + ant_{ht} - \frac{r^2}{2a_{ekst}}.$$

- c. For all other values of k , the grazing angle is determined as

$$\psi_i = \text{MAX}(0, \psi),$$

$$\psi = \psi_{ray_{k-1}} + \psi_{ray_k} \left[\frac{r - r_{ray_{k-1}}}{r_{ray_k} - r_{ray_{k-1}}} \right].$$

2. For range r greater than r_{hor} , an iterative loop is performed to determine the number of elements icr within array r_{ray} satisfying the condition $rmd < r_{ray} < rmd + \Delta r_{PE}$, where rmd is the range at mid-PE range step. The following steps a through b are then performed.

a. If icr is non-zero, then the indices $jr1$ and $jr2$ are initialized such that ranges $r_{ray_{jr1}}$ through $r_{ray_{jr2}}$ satisfies the condition in step 2 and $jr2 - jr1$ equals icr . If $r_{ray_{jr2}}$ is greater than range r , then the grazing angle Ψ_i is interpolated as

$$\Psi_i = \psi_{ray_j} + \psi_{ray_{j+1}} \left[\frac{r - r_{ray_j}}{r_{ray_{j+1}} - r_{ray_j}} \right]$$

where the index j lies between $jr1$ and $jr2$ and is defined such that r_{ray_j} is the nearest range point less than r and $r_{ray_{j+1}}$ is the nearest range point greater than r . If $r_{ray_{jr2}}$ is less than r , then the grazing angle Ψ_i is averaged according to

$$\Psi_i = \left(\frac{1}{icr} \right) \sum_{j=jr1}^{jr2} \psi_j$$

b. If icr is equal to 0 and r is greater than per then grazing angle Ψ_i is computed from interpolation of angles ψ_{PE} determined from spectral estimation. If no spectrally estimated angles exist, then Ψ_i is set equal to 0. If an evaporation duct profile has been specified, then Ψ_i is set equal to $\psi_{ray_{igrz}}$.

3. Both rmd and r are then incremented by Δr_{PE} .

Once all grazing angles Ψ have been determined, the arrays ψ_{ray} and ψ_{PE} are deallocated and the SU is exited.

Table 38 and Table 39 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the GRAZE_INT SU.

Table 38. GRAZE_INT SU input data element requirements.

Name	Description	Units	Source
a_{cut}	Tangent angle from antenna height to radio horizon	radians	PEINIT SU
a_{ekst}	$\frac{4}{3}$ times mean earth radius	meters	APM_MOD
ant_{ht}	Transmitting antenna height above local ground	meters	Calling CSCI
Δr_{grz}	PE range step used for calculation of grazing angles	meters	APMINIT CSC
Δr_{PE}	PE range step	meters	PEINIT SU
Δr_{PE2}	$\frac{1}{2}$ PE range step	meters	PEINIT SU
f_{ter}	Logical flag indicating if terrain profile has been specified: '.true.' = terrain profile specified '.false.' = terrain profile not specified	N/A	TERINIT SU
$hmref$	Height relative to h_{minter}	meters	TERINIT SU
$ihybrid$	Integer indicating which sub-models will be used: 0 = pure PE model 1 = full hybrid model (PE + FE + RO + XO) 2 = partial hybrid model (PE + XO)	N/A	APMINIT CSC
i_{gPE}	Number of grazing angles computed from spectral estimation	N/A	GETGRAZE SU
i_{grz}	Number of grazing angles computed from ray trace	N/A	GETGRAZE SU
i_{PE}	Number of PE range steps	N/A	PEINIT SU
l_{duct}	Logical flag indicating if surface-based duct profile has been specified '.true.' = surface-based duct exists '.false.' = no surface-based duct exists	N/A	REFINIT SU
l_{evap}	Logical flag indicating if evaporation duct profile has been specified '.true.' = evaporation duct exists '.false.' = no evaporation duct exists	N/A	REFINIT SU
ψ_{PE}	Array containing grazing angles computed from spectral estimation of PE field	radians	GETGRAZE SU
ψ_{ray}	2-dimensional array containing grazing angles and corresponding ranges r_{ray} computed from ray trace	radians, meters	GETGRAZE SU
r_{flat}	Maximum range at which the terrain profile remains flat from the source	meters	Calling SU
r_{hor}	Radio horizon range	meters	PEINIT SU
r_{max}	Maximum output range	meters	Calling CSCI
y_{fref}	Ground elevation height at the source	meters	APMINIT CSC

Table 39. GRAZE_INT SU output data element requirements.

Name	Description	Units
i_{error}	Integer variable indicating error number for ALLOCATE and DEALLOCATE statements	N/A
Ψ	Array of interpolated grazing angles at each PE range step	radians

5.1.16 Interpolate Profile (INTPROF) SU

The purpose of the INTPROF SU is to perform a linear interpolation vertically with height on the refractivity profile, $refref$. Interpolation is performed at each PE mesh height point.

In order to interpolate vertically at each PE mesh height, the following iteration is performed. The index j is determined such that for every i^{th} PE bin, ht_i is just greater than $href_j$ and $j < nlvl$. The interpolated profile $profint$ is then determined from

$$profint_i = refref_{j-1} + con \left(refref_j - refref_{j-1} \right) \frac{ht_i - href_{j-1}}{href_j - href_{j-1}}; \quad i = 1, 2, 3, \dots, n_{fft},$$

where the array ht and constant con have been determined in the APMINIT CSC.

Table 40 and Table 41 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the INTPROF SU.

Table 40. INTPROF SU input data element requirements.

Name	Description	Units	Source
con	$10^{-6} k_o$	meters ⁻¹	APMINIT CSC
$href$	Heights of refractivity profile with respect to local ground height	meters	PROFREF SU
ht	PE mesh height array of size n_{fft}	meters	PEINIT SU
n_{fft}	Transform size	N/A	FFTPAR SU
$nlvl$	Number of levels in new profile	N/A	PROFREF SU
$refref$	Refractivity array	M-units	PROFREF SU

Table 41. INTPROF SU output data element requirements.

Name	Description	Units
$profint$	Profile interpolated to every Δz_{PE} in height	M-units

5.1.17 PE Initialization (PEINIT) SU

The purpose of the PEINIT SU is to initialize all variables used in the PE model for subsequent calls to the PESTEP SU.

Upon entering the SU, several variables are initialized. The following PE transform variables are computed – the angle (or p-space) mesh size, Δp ; the Fourier transform normalization constant, f_{norm} ; the angle bin width, $\Delta\theta$, and various transform size factors:

$$\Delta p = \frac{\pi}{z_{max}}, \quad f_{norm} = \frac{2}{n_{fft}}, \quad n_{34} = \frac{3}{4}n_{fft},$$

$$\Delta\theta = \frac{\Delta p}{k_o}, \quad n_{ml} = n_{fft} - 1, \quad n_4 = \frac{1}{4}n_{fft},$$

The ALLARRAY_PE SU is then referenced to allocate and initialize all arrays associated with PE calculations.

Next, the horizon range, r_{hor} , for 0 receiver height and the tangent angle, a_{cut} , to the radio horizon are computed:

$$r_{hor} = 4121.81198 \sqrt{ant_{ht}}$$

$$a_{cut} = \text{TAN}^{-1} \left(\frac{ant_{ht}}{r_{hor}} \right) .$$

A temporary range step variable is computed as

$$\Delta rt = 55.67485 + 3.52969 \times 10^{-3} r_{max} - .01122 \times 10^{-6} r_{max}^2 .$$

Due to numerical constraints, limits will be imposed on the PE range step as follows. If performing a terrain case, then the PE range step is computed from

$$\Delta r_{PE} = \text{MAX} \left(\text{MIN} \left(2k_o \Delta z_{PE}^2, 700. \right), \Delta rt \right).$$

If r_{fix} (previously determined in the TERINIT SU) is greater than 0, then the temporary range step variable r_d is given by $r_d = \frac{r_{fix}}{\Delta r_{PE}}$, and Δr_{PE} is recomputed according to

$$\Delta r_{PE} = \mathbf{NINT}\left(\frac{1}{r_d}\right) r_{fix}; \text{ for } r_d < 1,$$

$$\Delta r_{PE} = \frac{r_{fix}}{\mathbf{NINT}(r_d)}; \text{ for } r_d \geq 1.$$

The variable iz_{inc} is then initialized to 1.

If no terrain profile is specified, then Δr_{PE} is given by

$$\Delta r_{PE} = \mathbf{MAX}(2k_o \Delta z_{PE}^2, \Delta r_t, 100.),$$

with the variable iz_{inc} initialized to 1 for frequencies greater than 10 GHz, 2 for frequencies greater than 5 GHz, and 3 otherwise.

If the PEINIT SU has been referenced from the GETGRAZE SU, then the PE range step (which in this case will be used for calculation of the grazing angle by spectral estimation) is further modified:

$$\Delta r_{PE} = \mathbf{MAX}(\Delta r_{PE}, 150.) - 1.$$

Otherwise, the range step is multiplied by the range step modifier r_{mult} .

The number of PE range steps is then computed:

$$i_{PE} = \mathbf{NINT}\left(\frac{r_{max}}{\Delta r_{PE}}\right).$$

If a terrain profile has been specified, the terrain elevations at each PE range step are now interpolated from the user-specified profile and stored in array tyh . The array is allocated for size i_{PE} , the range r is initialized to 0, and the elements in tyh are determined according to

$$tyh_i = ty_k + slp_k (r - tx_k); \quad i = 1, 2, \dots, i_{PE},$$

where r is $i\Delta r_{PE}$ and the index k is determined such that $tx_k < r \leq tx_{k+1}$.

The filter array, $filt$, for subsequent filtering of the PE field, is given by

$$filt_i = \frac{1}{2} + \frac{1}{2} \cos\left(i \frac{\pi}{n_4}\right); \quad i = 1, 2, \dots, n_4$$

The PE mesh height array ht is next given by

$$ht_i = i \Delta z_{PE}; \quad i = 1, 2, \dots, n_{fft}.$$

Next, the free-space propagator array $frsp$ is computed for subsequent use in the PESTEP SU. The propagator term is computed at each PE angle, or p-space, mesh point using the wide-angle propagator. A filter, or attenuation function (frequently called “window”), is then applied to the upper $\frac{1}{4}$ of the array corresponding to the highest $\frac{1}{4}$ of the maximum propagation angle.

The complex free-space propagator phase array $frsp$ is given by

$$frsp_j = f_{norm} e^{i \Delta r_{PE} \left(\sqrt{k_o^2 - (j \Delta p)^2} - k_o \right)}; \quad j = 0, 1, 2, \dots, n_{fft}.$$

where i is the imaginary number $\sqrt{-1}$. The upper $\frac{1}{4}$ of the free-space propagator array is filtered by a cosine-tapered (Tukey) filter array, $filt$ according to

$$frsp_j = filt_{j-n_{34}} frsp_j; \quad j = n_{34}, n_{34} + 1, n_{34} + 2, \dots, n_{fft}.$$

If a simple environmental case has been specified with no terrain and a range-independent refractivity profile, then the INTPROF SU is referenced for interpolation of the refractivity at every PE mesh point. The z-space propagator array $envpr$ is then computed from

$$envpr_j = e^{i \Delta r_{PE} profint_j}; \quad j = 0, 1, 2, \dots, n_{fft},$$

where i is the imaginary number $\sqrt{-1}$ and $profint$ is the sampled profile obtained from the INTPROF SU. The upper $\frac{1}{4}$ of $envpr$ is filtered by a cosine-tapered (Tukey) filter array, $filt$, according to

$$envpr_j = filt_{j-n_{34}} envpr_j; \text{ for } j = n_{34}, n_{34} + 1, n_{34} + 2, \dots, n_{fft}.$$

Finally, the XYINIT SU is referenced to determine the initial PE solution and the SU is exited.

Table 42 and Table 43 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the PEINIT SU.

Table 42. PEINIT SU input data element requirements.

Name	Description	Units	Source
ant_{ht}	Transmitting antenna height above local ground	meters	Calling CSCI
Δz_{PE}	Bin width in z-space	meters	FFTPAR SU
f_{MHz}	Frequency	MHz	Calling CSCI
f_{ter}	Logical flag indicating if terrain profile has been specified: ‘.true.’ = terrain profile specified ‘.false.’ = terrain profile not specified	N/A	TERINIT SU
i_{flag}	Integer flag indicating where in the APMINIT CSC the PEINIT SU is being referenced 0 = called before reference to GETGRAZE SU 1 = called for “real” PE run	N/A	Calling SU
i_{pl}	Polarization flag 0 = horizontal 1 = vertical	N/A	Calling SU
k_o	Free-space wave number	meters ⁻¹	APMINIT CSC
n_{fft}	Transform size	N/A	FFTPAR SU
n_{prof}	Number of refractivity profiles	N/A	Calling CSCI
r_{fix}	Fixed range increment of terrain profile	meters	Calling SU
r_{max}	Maximum specified range	meters	Calling CSCI
r_{mult}	PE range step multiplication factor	N/A	Calling CSCI
slp	Slope of each segment of terrain	N/A	TERINIT SU
tx	Range points of terrain profile	meters	TERINIT SU
ty	Adjusted height points of terrain profile	meters	TERINIT SU
z_{max}	Total height of the FFT/PE calculation domain	meters	FFTPAR SU

Table 43. PEINIT SU output data element requirements.

Name	Description	Units
a_{cut}	Tangent angle from antenna height to radio horizon	radians
Δr_{PE}	PE range step	meters
$\Delta \theta$	Angle bin width	radians
$envpr$	Complex [refractivity] phase term array interpolated every Δz_{PE} in height	N/A
$filt$	Cosine-tapered (Tukey) filter array	N/A
f_{norm}	Normalization factor	N/A
$frsp$	Complex free-space propagator term array	N/A
ht	PE mesh height array of size n_{fft}	meters

Table 43. PEINIT SU output data element requirements. (Continued)

Name	Description	Units
i_{error}	Integer variable indicating error number for ALLOCATE and DEALLOCATE statements	N/A
i_{PE}	Number of PE range steps	N/A
iz_{inc}	Integer increment for storing points at top of PE region (i.e., points are stored at every iz_{inc} range step)	N/A
n_{34}	$\frac{3}{4} n_{fft}$	N/A
n_4	$\frac{1}{4} n_{fft}$	N/A
n_{m1}	$n_{fft} - 1$	N/A
r_{hor}	Radio horizon range	meters
tyh	Adjusted height points of terrain profile at every PE range step.	meters
U	Complex PE field	$\mu\text{V/m}$

5.1.18 Profile Reference (PROFREF) SU

The purpose of the PROFREF SU is to adjust the current refractivity profile so that it is relative to a reference height, y_{ref} . The reference height is initially the minimum height of the terrain profile. Upon subsequent calls from the PESTEP SU, the refractivity profile is adjusted by the local ground height at each PE range step.

The reference height y_{ref} , depending on the value of i_{flag} , can be either h_{minter} or the local ground height above h_{minter} . If i_{flag} is 0, the profile arrays $refref$ and $href$ will be relative to h_{minter} and will also be used to initialize $refdum$ and $htdum$. If i_{flag} is 1, then the profile arrays $refref$ and $href$ will be referenced to the local ground height. The parameter h_{minter} is the reference height for internal calculations in the APM CSCI of the complex field U . Both arrays $refdum$ and $htdum$ are dummy arrays containing refractivity values and height values, respectively, for the currently interpolated profile.

The determination of $refref$ and $href$ proceeds as follows. First, the index $nlvl$ is initialized to the number of refractivity levels, $lvlep$, in $refdum$ and $htdum$; and $refref$ and $href$ are initialized to zero. Next, a test is made to determine whether the absolute value of the reference height y_{ref} is greater than 10^{-3} (i.e., is y_{ref} greater than approximately 0). If y_{ref} is approximately zero, the elements of $refref$ are set equal to the corresponding M-unit values of $refdum$, and the elements of $href$ are set equal to the corresponding height values of $htdum$ and the SU is exited.

For the case when y_{ref} is not zero, the following calculations are made. First, the flag i_{bmsl} and the index j_s are set equal to zero and minus one, respectively. Then, y_{ref} is tested to determine if it is below mean sea level. If so, i_{bmsl} and j are set equal to one and zero, respectively. If y_{ref} is not below mean sea level, then the refractivity profile level at which y_{ref} is just above is determined. The index j is determined such that $y_{ref} \leq htdum_{j+1}$ and $y_{ref} > htdum_j$.

The refractivity at y_{ref} is now computed from

$$rmu = refdum_j + (refdum_{j+1} - refdum_j) \frac{y_{ref} - htdum_j}{htdum_{j+1} - htdum_j}.$$

If y_{ref} falls below mean sea level and the extrapolation flag i_{extra} is zero, then rmu is given by

$$rmu = refdum_j + 0.118 \frac{y_{ref} - htdum_j}{htdum_{j+1} - htdum_j}.$$

The first element in $refref$ and $href$ is now set equal to rmu and 0, respectively. The number of refractivity levels in the arrays is now $l_{new} = nlvl - j$ and the remainder of the current refractivity profile is adjusted in height and stored in $refref$ and $href$ according to

$$\begin{aligned} refref_i &= refdum_k \\ href_i &= htdum_k - y_{ref}; \quad i = 1, 2, 3, \dots, l_{new}, \end{aligned}$$

where the index k is initialized to $j+1$ and is incremented by one with each iteration of i . The variable $nlvl$, indicating the number of levels in the newly created profile, is now set to l_{new} .

Finally, if i_{flag} equals zero, then $lvlep$ is set equal to $nlvl$ and $refref$ and $href$ are used to initialize $refdum$ and $htdum$, respectively, before exiting.

Table 44 and Table 45 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the PROFREF SU.

Table 44. PROFREF SU input data element requirements.

Name	Description	Units	Source
<i>htdum</i>	Height array for current interpolated profile	meters	REFINTER SU
<i>i_{extra}</i>	Extrapolation flag for refractivity profiles entered below mean sea level 0 = extrapolate to minimum terrain height standard atmosphere gradient 1 = extrapolate to minimum terrain height using first gradient in profile	N/A	Calling CSCI
<i>i_{flag}</i>	Integer flag indicating height at which to reference the refractivity profile 0 = adjust profile relative to h_{minter} 1 = adjust profile relative to local ground height above h_{minter}	N/A	Calling SU
<i>lvlep</i>	Number of height/refractivity levels in profile <i>refdum</i> and <i>htdum</i>	N/A	Calling CSCI
<i>refdum</i>	M-unit array for current interpolated profile	M-units	REFINTER
<i>y_{ref}</i>	Ground elevation height at current range	meters	Calling SU

Table 45. PROFREF SU output data element requirements.

Name	Description	Units
<i>href</i>	Height array for current interpolated profile	meters
<i>htdum</i>	Dummy array containing height values for current (horizontally interpolated) profile	meters
<i>lvlep</i>	Number of height/refractivity levels in profile	N/A
<i>nlvl</i>	Number of levels in new profile	N/A
<i>refdum</i>	M-unit array for current interpolated profile	M-units
<i>refref</i>	Refractivity array	M-units

5.1.19 RD Trace (RDTRACE) SU

The purpose of the RDTRACE SU is to perform ray traces of many rays launched within an angle of $\pm 4^\circ$. All angles from rays striking the surface are then sorted and stored for subsequent interpolation in the GRAZE_INT SU.

Upon entering the SU, four in-line ray trace functions are defined for general parameters a , b , c , and g_{rd} : RADA1, AP, RP, and HP. These function definitions are identical to those given in section 5.1.14.

Rays are traced with different angular increments at varying intervals. Angular increments are determined such that 1500 rays will be traced between the angular interval $\pm 0.5^\circ$, 1000 rays will be traced with launch angles between $|0.5^\circ|$ and $|\theta_r|$, and 500

rays will be traced for angles between $|\theta_t|$ and $|\vartheta_{mxg}|$. The angular increments are computed as

$$\begin{aligned} ainc_1 &= \frac{\theta_t}{1500}, \\ ainc_2 &= \frac{\theta_t}{500}, \\ ainc_3 &= \frac{\vartheta_{mxg} - \theta_t}{250}, \end{aligned}$$

where θ_t is 1° for i_{hybrid} equal to 1 and 1.5° otherwise. The maximum number of rays to trace, n_{ray} , is then initialized to a large value of 10 times the amount specified above.

Next, the grazing angle array ψ_{ray} is allocated and initialized with the first element set equal to $\frac{1}{2}\pi$. The height of the terrain y_t at the current traced range step is initialized to 0, or tyh_l if a terrain profile has been specified ($f_{ter} = \text{'true.'}$). The number of grazing angles i_{grz} and the launch angle a_{launch} are initialized to 0 and $-\vartheta_{mxg}$, respectively. A DO loop is now implemented where the following steps 1 through 4 are performed an n_{ray} number of times.

1. At the start of the ray trace, the current local angle (a_0), range (r_0), height (h_0), and refractive gradient index (j) are initialized to a_{launch} , 0, ant_{ref} , and i_{start1} , respectively. The terrain slope t_{slope} of the terrain segment at the current traced range is also initialized to 0.
2. If the antenna height ant_{ref} is at an inflection point within the refractivity profile, then the following steps 2.a through 2.c are performed, otherwise the current range to trace to, ro , is initialized to 0 and the SU proceeds with step 3.
 - a. If a_0 is near zero (i.e., $|a_0|$ is less than 10^{-6}) and no terrain has been specified ($f_{ter} = \text{'false.'}$), then the grazing angle and range stored is 0 and r_{max} , respectively. The counter i_{grz} is incremented by 1, the launch angle is incremented by $ainc_1$, and the SU then repeats steps 1 through 4.
 - b. If a_0 is near zero and a terrain profile has been specified, then a check is made to determine if the ray intersects a segment of the terrain profile downrange. If so, the range at the point of reflection r_1 and the grazing angle are computed as

$$h = \frac{h_0 - tyh_{k-1}}{tyh_k - tyh_{k-1}}$$

$$r_1 = [(k-1) + h]\Delta r_{PE}$$

$$\psi = \mathbf{TAN}^{-1}\left(\frac{tyh_k - tyh_{k-1}}{\Delta r_{PE}}\right),$$

where k is determined such that $tyh_{k-1} < h_0 < tyh_k$. The reflection range and grazing angle are then stored, the counter i_{grz} is incremented by 1, and the launch angle is incremented by $ainc_1$. The SU then repeats steps 1 through 4.

- c. If a_0 is less than 0, the index j is decremented by 1 and the SU proceeds with step 3.
3. A loop is now begun to trace a ray starting with launch angle a_{launch} to every PE range step. The current range to trace to, ro , is incremented by Δr_{PE} , and the following steps 3.a through 3.h are performed until one of the following conditions are met: r_0 reaches ro , h_0 reaches ht_{lim} , or the difference in reflection range between consecutive grazing angles is less than 10^{-3} . All references to the index i in the steps below refer to the index in this loop varying from 1 to the number of PE range steps, i_{PE} .
 - a. The range r_1 at the end of the range step and the refractive gradient g_{rd} within the current range step are initialized to ro and $grdum_j$, respectively. The propagation angle a_1 at the end of the range step is computed by $\mathbf{AP}(a_0, r_1 - r_0)$.
 - b. If $|a_0|$ is less than 10° , then if a_0 and a_1 differ in sign, a_1 is set equal to 0 and r_1 is computed from $\mathbf{RP}(r_0, a_1 - a_0)$. The height of the ray at the end of the range step is next computed by $\mathbf{HP}(h_0, a_1, a_0)$. If $|a_0|$ is greater than 10° , then a_1 is set equal to a_0 and h_1 is computed as

$$h_1 = h_0 + (r_1 - r_0)\mathbf{TAN}^{-1}(a_1) - \frac{(r_1 - r_0)^2}{twoka},$$

where $twoka$ is determined in the GET_K SU.

- c. Once r_1 , h_1 , and a_1 have been computed, it must be determined if the height of the ray has fallen below the elevation height of the terrain at the current range step. The following steps 3.c.i through 3.c.iii are performed if a terrain profile has been specified. Otherwise, the SU proceeds with step 3.d.
 - i. The height h_{ter} of the terrain at range r_1 is computed as

$$t_{slope} = \frac{tyh_i - tyh_{i-1}}{\Delta r_{PE}},$$

$$h_{ter} = tyh_{i-1} + [r_1 - (i-1)\Delta r_{PE}]t_{slope}$$

- ii. If h_1 is less than h_{ter} , then the ray has a reflection point and the range and height of reflection is determined by

$$r_{slope} = \frac{h_1 - h_0}{r_1 - r_0}, \quad \text{with } r_1 \text{ recomputed as}$$

$$r_1 = \frac{tyh_{i-1} - h_0 - t_{slope}(i-1)\Delta r_{PE} + r_{slope}r_0}{r_{slope}t_{slope}},$$

$$h_1 = h_0 + r_{slope}(r_1 - r_0).$$

- iii. If a new range r_1 and height h_1 has been computed, then if $|a_0|$ is less than 10° , then a_1 is recomputed using **AP**($a_0, r_1 - r_0$). If $|a_0|$ is greater than 10° , then a_1 is set equal to a_0 and h_1 is recomputed as

$$h_1 = h_0 + (r_1 - r_0)\mathbf{TAN}^{-1}(a_1) - \frac{(r_1 - r_0)^2}{twoka}.$$

- d. Next, it must be determined if the ray has passed through a refractive layer, in which case the index must be adjusted and the range, height, and angle must be computed at the refractive layer transition. For an upward ray, if a_1 is positive and h_1 has reached or surpassed the next height level, then a_1, r_1, j , and h_1 , are found as follows. First, h_1 is set equal to $htdum_{j+1}$, j is increment by 1, and a_1 and r_1 are given by

$$a_1 = \sqrt{\mathbf{RADA1}(a_0, h_1 - h_0)}$$

$$r_1 = \mathbf{RP}(r_0, a_1 - a_0)$$

- e. For a downgoing ray, if a_1 is less than or equal to 0, and h_1 is less than $htdum_j$, then h_1 is set equal to $htdum_j$, and j is set equal to the maximum of 0 or $j-1$. The variables a_1 and r_1 are then determined from

$$a_1 = -\sqrt{\mathbf{RADA1}(a_0, h_1 - h_0)}$$

$$r_1 = \mathbf{RP}(r_0, a_1 - a_0)$$

- f. If the ray has hit the surface and is reflected, which would be the condition for which the index j is equal to 0, then the grazing angle and the angle of reflection with respect to the horizontal are computed as

$$\begin{aligned}\psi &= \text{TAN}^{-1}(t_{slope}) - a_1 \\ a_{ref} &= 2 \text{TAN}^{-1}(t_{slope}) - a_1\end{aligned}$$

The range r_1 and grazing angle are then stored for later use in the GRAZE_INT SU.

- g. In preparation for the next ray trace step, h_0 is set equal to h_1 , r_0 is set equal to r_1 , and a_0 is set equal to a_{ref} . If a_0 is greater than $\frac{1}{2}\pi$, or if r_1 has reached ro , then the current iteration is exited and the SU proceeds to step 3.h; otherwise, steps 3.a through 3.g are repeated until r_0 reaches ro .
- h. The range ro [to trace to] for the next step is incremented by Δr_{PE} and step 3 is repeated for all range steps.
4. Once a ray has been traced through the entire i_{PE} number of range steps, a_{launch} is increased. If $|a_{launch}|$ is less than 0.5° , the launch angle is increased by a_{inc1} ; otherwise, if $|a_{launch}|$ is less than θ_t , then it is increased by a_{inc2} . If neither of these conditions are met, then a_{launch} is increased by a_{inc3} . Steps 1 through 4 are repeated until an n_{ray} number of rays have been traced.

Finally, the grazing angles are sorted by range and stored in array ψ_{ray} and the SU is exited.

Table 46 and Table 47 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the RDTRACE SU.

Table 46. RDTRACE SU input data element requirements.

Name	Description	Units	Source
ant_{ref}	Transmitting antenna height relative to the reference height h_{minter}	meters	TERINIT SU
Δr_{PE}	PE range step	meters	PEINIT SU
f_{ter}	Logical flag indicating if terrain profile has been specified: ‘.true.’ = terrain profile specified ‘.false.’ = terrain profile not specified	N/A	TERINIT SU
$grdum$	Array of refractivity gradients defined by profile $htdum$ and $refdum$	M-units/ meter	REFINTER SU
h_{max}	Maximum output height with respect to mean sea level	meters	Calling CSCI
$htdum$	Height array for current interpolated profile	meters	REFINTER SU
ht_{lim}	User-supplied maximum height relative to h_{minter} , i.e., $ht_{lim} = h_{max} - h_{minter}$	meters	TERINIT SU
i_{hybrid}	Integer indicating which sub-models will be used: 0 = pure PE model 1 = full hybrid model (PE + FE + RO + XO) 2 = partial hybrid model (PE + XO)	N/A	APMINIT CSC
i_{PE}	Number of PE range steps	N/A	PEINIT SU
i_{start1}	Refractivity level index within $htdum$ at ant_{ref}	N/A	REFINIT SU
$refdum$	M-unit array for current interpolated profile	M-units	REFINTER
r_{max}	Maximum output range	meters	Calling CSCI
ϑ_{mxg}	Maximum PE calculation angle for spectral estimation of grazing angles	radians	APMINIT CSC
$twoka$	Twice the effective earth radius	meters	GET_K SU
tyh	Adjusted height points of sampled terrain profile at every PE range step	meters	TERINIT SU

Table 47. RDTRACE SU output data elements requirements.

Name	Description	Units
i_{grz}	Number of grazing angles computed from ray trace	N/A
ψ_{ray}	2-dimensional array containing grazing angles and corresponding ranges computed from ray trace	radians, meters

5.1.20 Refractivity Initialization (REFINIT) SU

The purpose of the REFINIT SU is to check for valid environmental profile inputs and to initialize all refractivity arrays used within one application of APM.

Upon entering, the maximum height h_{large} at which the refractivity profile is extrapolated is set to 10^6 meters in a DATA statement. In addition, i_{error} is initialized to zero.

The environmental data is checked to determine if range-dependent profiles have been specified ($n_{prof} > 1$). If so, the range of the last profile entered, $rngprof_{n_{prof}}$ is checked and if it is less than the maximum output range specified, r_{max} , an error message is returned (i.e., i_{error} is set equal to -12) depending on the value of error flag, $lerr12$, set in the NITES application itself. The SU is then exited; otherwise, if no error occurs, the SU proceeds to the next step.

Next, the REFINIT SU tests for valid refractivity level entries for each profile. Every user-specified profile is tested to make sure the first level in the profile begins with a value of zero height (or less than zero if the first level is below mean sea level). If it does not, i_{error} is set to -13 and the SU is exited; otherwise, the SU proceeds to the next step.

A test is then made to determine if the last gradient in each profile is negative. If the last gradient in any profile is negative, i_{error} is set to minus fourteen and the SU is exited; otherwise, an additional refractivity level is extrapolated to height h_{large} and added to each profile. The additional level is added according to

$$\begin{aligned} hmsl_{lvl,i} &= h_{large}, \\ refmsl_{lvl,i} &= refmsl_{lvl-1,i} + grd[h_{large} - hmsl_{lvl-1,i}] \end{aligned} \quad i = 1, 2, 3, \dots, n_{prof}$$

where

$$grd = \frac{refmsl_{lvl-1,i} - refmsl_{lvl-2,i}}{hmsl_{lvl-1,i} - hmsl_{lvl-2,i}}.$$

The counter for the current profile, i_s , is now initialized to 1 and the range of the next refractivity profile, rv_2 , is initialized to $rngprof_{i_s}$. Next, the results of the extrapolation of the first environmental profile (i.e., the profile at range 0) are transferred to dummy arrays, $htdum$ and $refdum$, respectively. The index $lvlp$ is now set equal to $lvlp$. Duplicate levels in the first profile are removed by a reference to the REMDUP SU, and $refdum$ and $htdum$ are adjusted to the minimum terrain height by a reference to the PROFREF SU. The parameter nvl , returned from the PROFREF SU, is now the number of height/refractivity levels in the adjusted $htdum$ and $refdum$ arrays.

If troposcatter calculations have been specified ($T_{rope} = \text{'true.'}$), then the surface refractivity at the transmitter $snref_{tx}$ is determined by referencing the PROFREF SU to adjust the profile relative to y_{ref} and initializing $snref_{tx}$ to $refref_0$.

Next, the height and thickness of the highest trapping layer (if one exists), h_{trap} and h_{thick} , respectively, are found relative to h_{minter} . First, h_{trap} and h_{thick} are initialized to zero. Then the following steps 1 through 2 are performed for each i^{th} profile and for each j^{th} refractivity level.

1. The gradient of the current height/refractivity level grd and its height relative to h_{minter} , h_{p1} , are found from

$$\begin{aligned} grd &= refmsl_{j+1,i} - refmsl_{j,i} \\ h_{p1} &= hmsl_{j+1,i} - h_{minter} \end{aligned} .$$

2. If grd is negative and h_{p1} is greater than h_{trap} , then h_{trap} is set equal to h_{p1} , and h_{p0} and h_{thick} are determined from

$$\begin{aligned} h_{p0} &= hmsl_{j,i} - h_{minter} \\ h_{thick} &= h_{p1} - h_{p0} \end{aligned} .$$

Next, the index level i_{start1} within the refractivity profile of the antenna height ant_{ref} is determined and the gradient array $grdum$ is computed as

$$grdum_i = 10^{-6} \left(\frac{refdum_{i+1} - refdum_i}{htdum_{i+1} - htdum_i} \right); \quad i = 0, 1, 2, \dots, nlvl - 1 .$$

If using the full hybrid mode ($i_{hybrid} = 1$), then the following steps 1 through 4 are performed to build arrays associated with ray optics (RO) calculations. Otherwise, the M-unit value rm_{tx} at the antenna height is determined from

$$rm_{tx} = 10^{-6} \left[refdum_{i_{start1}} + grdum_{i_{start1}} 10^6 (ant_{ref} - htdum_{i_{start1}}) \right]$$

and the SU continues with the procedures following step 4.

1. First, the refractivity and height arrays rm and zrt are built. All elements in zrt are set equal to all elements in $htdum$. An additional height level, equal to ant_{ref} , is included in zrt and the index i_{start} is initialized to that height level which corresponds to ant_{ref} . Array rm is given by

$$rm_i = 10^{-6} refdum_i, \quad i = 1, 2, 3, \dots, nlvl ,$$

with the refractivity level at height ant_{ref} interpolated according to

$$rm_{i_{start}} = rm_{i_{start}+1} + \left(ant_{ref} - zrt_{i_{start}-1} \right) \left(\frac{rm_{i_{start}+1} - rm_{i_{start}-1}}{zrt_{i_{start}+1} - zrt_{i_{start}-1}} \right).$$

The total number of levels *levels* in *zrt* is reduced by 1 since the highest level is not needed.

2. For the special case when the terrain profile is initially flat, but at non-zero height, the following steps 2.a through 2.d are performed to adjust the refractivity arrays *rm* and *zrt* associated with RO calculations. First, the index *nlevel* is initialized to the number of refractivity levels, *levels*; *y_{ref}* is initialized to *ty₁*; *refref* and *href* are initialized to zero; and the index *js* is initialized to -1.

- a. Next, *js* is determined such that $zrt_{js} < y_{ref} \leq zrt_{js+1}$. If a value for *js* is not found such that this condition holds true (i.e., *js* remains at -1), then the SU proceeds with step 2.d.

- b. The refractivity at *y_{ref}* is now computed from

$$f_{rac} = \frac{y_{ref} - zrt_{js}}{zrt_{js+1} - zrt_{js}},$$

$$rmu = rm_{js} + f_{rac}(rm_{js+1} - rm_{js}).$$

If $\text{INT}(f_{rac})$ is equal to 1, then *js* is set equal to *js*+1. The temporary counter *l_{new}* is initialized to *nlevel*-*js*.

- c. The first element in *refref* and *href* is now set equal to *rmu* and 0, respectively. The remainder of the current refractivity profile is adjusted in height and stored in *refref* and *href* according to

$$refref_j = rm_k$$

$$href_j = zrt_k - y_{ref}; \quad j = 1, 2, 3, \dots, l_{new},$$

where the index *k* is initialized to *js*+1 at the start and is incremented by one with each iteration of *j*. The variable *levels*, indicating the number of levels in the newly created profile, is now set to *l_{new}*. *refref* and *href* are now used to initialize *rm* and *zrt*.

- d. The variable *i_{start}* is now reduced by the amount *js*.

3. The arrays gr and q , used in RO and ray-tracing calculations, are determined next. The gradient array gr is given by

$$gr_i = \frac{rm_{i+1} - rm_i}{zrt_{i+1} - zrt_i}; \quad i = 0, 1, 2, \dots, levels,$$

The array q is given by

$$q_i = 2(rm_{i+1} - rm_i); \quad i = 0, 1, 2, \dots, levels.$$

4. The M-unit value rm_{tx} at the antenna height is now set equal to $rm_{i_{start}}$.

Next, the minimum M-unit value rm_{min} of the refractivity at range 0 is determined by searching for the minimum numerical value in array $refdum$ and assigning rm_{min} this value. The maximum M-unit value rm_{max} at or below the antenna height is then determined from

$$rm_{max} = \mathbf{MAX}(10^6 rm_{tx}, refdum_i), \quad i = 0, 1, 2, \dots, i_{start1}.$$

Both rm_{min} and rm_{max} are then multiplied by 10^{-6} . If the antenna is within a duct, the flag l_{duct} is set to '.true.', and the critical angle a_{crit} is computed as

$$a_{crit} = \sqrt{2(rm_{tx} - r_{crit})} + 10^{-6},$$

where r_{crit} is the minimum M-unit value in the profile for levels above the height ant_{ref} .

Finally, a check is made to determine if an evaporation profile exists. This check is performed only if a range-independent profile has been specified ($n_{prof} = 1$) and if h_{trap} is greater than 0. An evaporation duct is assumed to exist if the height at which rm_{min} occurs is less than 40 meters and if the difference between successive gradients is less than or equal to 0.1. If these two conditions occur, then the flag l_{evap} is set to '.true.'.

Table 48 and Table 49 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the REFINIT SU.

Table 48. REFINIT SU input data element requirements.

Name	Description	Units	Source
ant_{ref}	Transmitting antenna height relative to the reference height h_{minter}	meters	TERINIT SU
f_{ter}	Logical flag indicating if terrain profile has been specified: 'true.' = terrain profile specified 'false.' = terrain profile not specified	N/A	TERINIT SU
h_{minter}	Minimum height of terrain profile	meters	TERINIT SU
$hmsl$	2-dimensional array containing heights with respect to mean sea level of each profile. Array format must be $hmsl_{i,j}$ = height of i^{th} level of j^{th} profile; $j=1$ for range-independent cases	meters	Calling CSCI
i_{hybrid}	Integer indicating which sub-models will be used: 0 = pure PE model 1 = full hybrid model (PE + FE + RO + XO) 2 = partial hybrid model (PE + XO)	N/A	APMINIT CSC
$lerr12$	User-provided error flag that will trap on certain errors if set to 'true.'	N/A	Calling CSCI
$lvlp$	Number of height/refractivity levels in profiles	N/A	Calling CSCI
$nprof$	Number of refractivity profiles	N/A	Calling CSCI
$refmsl$	2-dimensional array containing refractivity with respect to mean sea level of each profile. Array format must be $refmsl_{i,j}$ = M-unit at i^{th} level of j^{th} profile; $j=1$ for range-independent cases	M-unit	Calling CSCI
r_{max}	Maximum range	meters	Calling CSCI
$rngprof$	Ranges of each profile. $rngprof_i$ = range of i^{th} profile	meters	Calling CSCI
T_{ropo}	Troposcatter calculation flag: 'false.' = no troposcatter calcs 'true.' = troposcatter calcs	N/A	Calling CSCI
ty	Adjusted height points of terrain profile	meters	TERINIT SU
y_{fref}	Ground elevation height at the source	meters	APMINIT CSC

Table 49. REFINIT SU output data element requirements.

Name	Description	Units
a_{crit}	Critical angle	radians
gr	Intermediate M-unit gradient array, RO region	(M-unit/m) 10^{-6}
$grdum$	Array of refractivity gradients defined by profile $htdum$ and $refdum$	M-units/meter
$hmsl$	2-dimensional array containing heights with respect to mean sea level of each profile. Array format must be $hmsl_{i,j}$ = height of i^{th} level of j^{th} profile; $j=1$ for range-independent cases	meters
$htdum$	Height array for current interpolated profile	meters
h_{thick}	Thickness of highest trapping layer from all refractivity profiles	meters

Table 49. REFINIT SU output data element requirements. (Continued)

Name	Description	Units
h_{trap}	Height of highest trapping layer from all refractivity profiles	meters
i_{error}	Integer value that is returned if any errors exist in input data	N/A
i_s	Counter for current profile	N/A
i_{start}	RO height index at antenna height	N/A
i_{start1}	Refractivity level index within $htdum$ at ant_{ref}	N/A
l_{duct}	Logical flag indicating if surface-based duct profile has been specified 'true' = surface-based duct exists 'false' = no surface-based duct exists	N/A
l_{evap}	Logical flag indicating if evaporation duct profile has been specified 'true' = evaporation duct exists 'false' = no evaporation duct exists	N/A
$levels$	Number of levels defined in zrt , rm , q , and gr arrays	N/A
$lvlep$	Number of height/refractivity levels in profile $htdum$, $refdum$	N/A
$lvlp$	Number of user-specified levels in refractivity profile (for range dependent case all profiles must have same number of levels)	N/A
$nlvl$	Number of height/refractivity levels in profile $refref$, $href$	N/A
q	Intermediate M-unit difference array, RO region	M-unit 10^{-6}
$refdum$	M-unit array for current profile	M-unit
$refmsl$	2-dimensional array containing refractivity with respect to mean sea level of each profile. Array format must be $refmsl_{ij}$ = M-unit at i^{th} level of j^{th} profile; $j=1$ for range-independent cases	M-unit
rm	Intermediate M-unit array, RO region	M 10^{-6}
rm_{max}	Maximum M-unit value of refractivity profile at range 0	meters
rm_{min}	Minimum M-unit value of refractivity profile at range 0	meters
rm_{tx}	M-unit value at height ant_{ref}	meters
$rv2$	Range of the next refractivity profile	meters
$snref_{tx}$	Surface refractivity at transmitter	M-unit
zrt	Intermediate height array, RO region	meters

5.1.21 Remove Duplicate Refractivity Levels (REMDUP) SU

The purpose of the REMDUP SU is to remove any duplicate refractivity levels in the current interpolated profile. Adjoining profile levels are checked to see if the heights are within 0.001 meters. If they are, the duplicate level in the profile is removed. This process continues until all profile levels ($lvlep$) have been checked.

Table 50 and Table 51 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the REMDUP SU.

Table 50. REMDUP SU input data element requirements.

Name	Description	Units	Source
<i>htdum</i>	Height array for current interpolated profile	meters	REFINIT SU REFINTER SU
<i>lvlep</i>	Number of height/refractivity levels in profile	N/A	REFINIT SU REFINTER SU
<i>refdum</i>	M-unit array for current interpolated profile	M-unit	REFINIT SU REFINTER SU

Table 51. REMDUP SU output data element requirements.

Name	Description	Units
<i>htdum</i>	Height array for current interpolated profile	meters
<i>lvlep</i>	Number of height/refractivity levels in profile	N/A
<i>refdum</i>	M-unit array for current interpolated profile	M-unit

5.1.22 Terrain Initialization (TERINIT) SU

The purpose of the TERINIT SU is to examine and initialize terrain arrays for subsequent use in PE calculations. It tests for and determines a range increment if it is found that range/height points are provided in fixed range increments. The minimum terrain height is determined, and the entire terrain profile is adjusted in height so that all internal calculations are referenced to this height. This is done in order to maximize the PE transform calculation volume.

First, several variables are initialized. The logical flag, f_{ter} , used to indicate whether the application at hand is a terrain case, is set equal to ‘.false.’ The integer flag, i_{error} , that is returned if any errors exist in input data, is set equal to zero. The maximum tangent ray angle, α_{it} , from source to terrain peak along the profile path is set equal to zero. The minimum height of the terrain profile, h_{minter} , is set equal to zero. The transmitting antenna height, ant_{ref} , relative to the reference height h_{minter} is set equal to ant_{ht} . The maximum terrain height, h_{termax} , along the profile path is set equal to zero. Finally, if the number of terrain points i_{tp} specified is greater than zero, then f_{ter} is set equal to ‘.true.’.

If performing a terrain case ($f_{ter} = \text{‘.true.’}$), the following steps (1 through 9) are performed; otherwise, the SU proceeds to step 10.

1. First, all terrain range points are checked in array *terx* to ensure they are steadily increasing. If they are not, the error flag i_{error} is set equal to -17 and the SU is exited. Otherwise, the SU proceeds to step 2.

2. Next, a test is made to determine whether the first range value is zero. If it is not, the error flag i_{error} is set equal to -18 and the SU is exited. Otherwise, the SU proceeds to step 4.
3. Next, a test is made to determine if the last range point within the terrain profile meets or exceeds r_{max} . If the logical flag $lerr6$ is 'true.' and if the condition $terx_{i_{tp}} < r_{max}$ is met, then i_{error} is set equal to -6 and the SU is exited; otherwise, the SU proceeds to step 4.
4. A check is now made to determine if the specified terrain range points are spaced at fixed increments. In this procedure, three variables, $rdif_1$, r_{frac} , and r_{difsum} are initialized to $terx_2 - terx_1$, zero, and $rdif_1$, respectively. The variable $rdif_1$ is the difference between adjacent terrain point ranges. The variable r_{frac} is the ratio between adjacent terrain point differences. The variable r_{difsum} is the running sum of adjacent terrain point differences. The final value for r_{difsum} and maximum r_{frac} are determined as

$$rdif_2 = \text{MAX}(10^{-3}, terx_{i+1} - terx_i); \quad i = 2, 3, 4, \dots, i_{tp} - 1$$

$$r_{frac} = \frac{rdif_2}{rdif_1}$$

$$r_{difsum} = r_{difsum} + rdif_2$$

where $rdif_1$ is set equal to the previous value of $rdif_2$ before each subsequent calculation of a new $rdif_2$, and r_{frac} is the maximum of all ratios computed.

5. If it is determined that the terrain points are spaced at fixed range increments, then the range spacing r_{fix} is set to this increment. Assuming that the range points are not equally spaced, r_{fix} is initially set equal to zero. If the value of r_{frac} is less than 1.05, then r_{fix} is determined from

$$r_{fix} = \text{NINT} \left(\frac{r_{difsum}}{i_{tp} - 1} \right) .$$

6. The minimum height h_{minter} of the terrain profile is now found and the entire terrain profile is adjusted by h_{minter} such that this is the new zero reference. The adjusted terrain profile is stored in arrays tx and ty (i.e., $ty = tery - h_{minter}$ for all elements in $tery$). Next, the maximum height h_{termax} of the terrain is also obtained from $tery$. If h_{termax} exceeds h_{max} , then i_{error} is set equal to -8 and the SU is exited. Otherwise, the SU proceeds with step 7.

7. An extra point is added to the arrays tx and ty . If $tx_{i_{tp}}$ is less than r_{max} , then $tx_{i_{tpa}}$ is set equal to r_{max} times 1.1. The input index i_{tpa} is the number of terrain points used internally in arrays tx and ty . If $tx_{i_{tp}}$ is greater or equal to r_{max} , then $tx_{i_{tpa}}$ is set equal to $tx_{i_{tp}}$ times 1.1. Finally, the array element $ty_{i_{tpa}}$ is set equal to $ty_{i_{tp}}$.
8. The variable ant_{ref} is set equal to ant_{ht} plus ty_1 . Next, the array of terrain slopes, slp , and the maximum tangent ray angle, α_u , from the source to the terrain peak along the profile path are found as follows. The slope, slp_i , for each i^{th} terrain segment is given by

$$slp_i = \frac{ty_{i+1} - ty_i}{\text{MAX}(tx_{i+1} - tx_i, 10^{-5})}; \quad i = 1, 2, 3, \dots, i_{tpa} - 1.$$

If the current slope is greater than the slope tolerance of 10^{-5} then it is assumed that the terrain profile is no longer flat at the current range and the variable r_{flat} is set equal to tx_i . Next, if the value of ty_i is greater than ant_{ref} , then the maximum tangent angle α_u from the source to each terrain point is calculated as

$$\alpha_u = \text{MAX} \left[\text{TAN}^{-1} \left(\frac{ty_i - ant_{ref}}{tx_i} \right) \right]; \quad i = 1, 2, 3, \dots, i_{tpa} - 1.$$

After α_u is determined, 0.5° is added to its value.

9. If rough surface calculations are required ($ruf = \text{'true'}$.) then a search of the specified ground types is performed to check if types are other than sea water. If ground types are other than sea water for the entire range of the terrain profile then rough surface calculations are turned off by setting ruf equal to 'false' (the APM CSCI only accounts for rough sea surface effects).
10. Before exiting, the minimum height hm_{ref} relative to h_{minter} is found from the difference between the minimum specified output height h_{min} and h_{minter} . The maximum height limit ht_{lim} relative to h_{minter} is given by the difference between h_{max} and h_{minter} . If the antenna height ant_{ref} is greater than ht_{lim} , the error code i_{error} is set to -9.

Table 52 and Table 53 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the TERINIT SU.

Table 52. TERINIT SU input data element requirements.

Name	Description	Units	Source
ant_{ht}	Transmitting antenna height above local ground	meters	Calling CSCI
$dielec$	2-dimensional array containing the relative permittivity and conductivity; $dielec_{1,i}$ and $dielec_{2,i}$, respectively.	N/A, S/m	Calling CSCI, DIEINIT SU
h_{max}	Maximum output height with respect to mean sea level	meters	Calling CSCI
h_{min}	Minimum output height with respect to mean sea level	meters	Calling CSCI
$igrnd$	Integer array containing ground type composition for given terrain profile - can vary with range. Different ground types are: 0 = sea water 1 = fresh water 2 = wet ground 3 = medium dry ground 4 = very dry ground 5 = ice at -1 degree C 6 = ice at -10 degree C 7 = user defined (in which case, values of relative permittivity and conductivity must be given).	N/A	Calling CSCI
i_{tp}	Number of height/range points in profile	N/A	Calling CSCI
i_{tpa}	Number of height/range points pairs in profile tx , ty	N/A	APMINIT CSC
$lerr6$	User-provided error flag that will trap on certain errors if set to '.true.'	N/A	Calling CSCI
r_{max}	Maximum output range	meters	Calling CSCI
ruf	Logical flag indicating if rough sea surface calculations are required .true. = perform rough sea surface calculations .false. = do not perform rough sea surface calculations	N/A	APMINIT CSC
$terx$	Range points of terrain profile	meters	Calling CSCI
$tery$	Height points of terrain profile	meters	Calling CSCI

Table 53. TERINIT SU output data element requirements.

Name	Description	Units
α_u	Maximum tangent ray angle from the source to the terrain peak along profile height	radians
ant_{ref}	Transmitting antenna height relative to the reference height h_{minter}	meters
f_{ter}	Logical flag indicating if terrain profile has been specified: .true. = terrain profile specified .false. = terrain profile not specified	N/A
h_{minter}	Minimum height of terrain profile	meters
hm_{ref}	Height relative to h_{minter}	meters
ht_{lim}	User-supplied maximum height relative to h_{minter} , i.e., $ht_{lim} = h_{max} - h_{minter}$	meters

Table 53. TERINIT SU output data element requirements. (Continued)

Name	Description	Units
h_{termax}	Maximum terrain height along profile path	meters
i_{error}	Integer value that is returned if errors exist in input data	N/A
r_{fix}	Fixed range increment of terrain profile	meters
slp	Slope of each segment of terrain	N/A
tx	Range points of terrain profile	meters
ty	Adjusted height points of terrain profile	meters

5.1.23 Trace to Output Range (TRACE_ROUT) SU

The purpose of the TRACE_ROUT SU is to trace a single ray, whose launch angle is specified by the calling routine, to each output range. The height of this ray is stored at each output range for subsequent proper indexing and accessing of the appropriate sub-models.

Upon entering the SU, four in-line ray trace functions are defined for general parameters a , b , c , and g_{rd} : RADA1, AP, RP, and HP. These function definitions are identical to those given in section 5.1.14.

Next, the propagation angle, range, and height at the beginning of the range step, a_0 , r_0 , h_0 , respectively, are initialized to a_s , r_s , and h_s – the angle, range and height specified by the calling SU. The profile index j is also initialized to j_s from the calling SU. The index j_r is determined such that $rngout_{j_r}$ is the first range point greater than r_0 . The array $harray$, containing the heights of the traced ray at each output range, is then set equal to 0 for elements 1 through j_r . The following steps 1 through 2 are performed for each output range step i from j_r to n_{rout} .

1. First, $harray_{j_r}$ is set equal to zero and the variable ro is set equal to $rngout_{j_r}$. Next, the following steps 1.a through 1.d are performed until r_0 has reached ro or h_0 has reached ht_{lim} .
 - a. First, the range, r_1 , at the end of the ray trace segment is set equal to ro . Then the current gradient g_{rd} is set equal to $grdum_j$. The angle, a_1 , at the end of the ray trace segment is found from AP(a_0 , r_1-r_0).
 - b. If a_1 is of the opposite sign as a_0 , then a_1 is set equal to zero and r_1 is given by RP(r_0 , a_1-a_0). h_1 is then given by HP(h_0 , a_1 , a_0).

- c. Now the value of h_1 is tested. If the value of h_1 is greater than or equal to $htdum_{j+1}$ then h_1 is set equal to the minimum of ht_{lim} or $htdum_{j+1}$, and a_1 , r_1 and the index j are re-computed as

$$a_1 = \sqrt{\mathbf{RADA}1(a_0, h_1 - h_0)}$$

$$r_1 = \mathbf{RP}(r_0, a_1 - a_0)$$

$$j = \mathbf{MIN}(lvlep, j + 1).$$

- d. If h_1 is less than $htdum_{j+1}$ but greater than ht_{lim} , then h_1 is set equal to ht_{lim} and a_1 and r_1 are re-computed as in step 1.c above. The angle, range, and height variables a_0 , r_0 , and h_0 are now set equal to a_1 , r_1 , and h_1 in preparation for the next step. Steps 1.a through 1.d are repeated until $r_0 \geq ro$.
2. Once r_0 has reached ro , $harray_{j_r}$ is then set equal to h_0 . The index j_r is then incremented by 1 and steps 1 through 2 are repeated for all output range steps or until h_0 has reached ht_{lim} .

Finally, if the traced ray has reached ht_{lim} at a range before r_{max} , then $harray$ is set equal to ht_{lim} for elements from j_r to n_{rout} , with the index i_{hmx} , indicating the element in $harray$ where this occurs, set equal to j_r .

Table 54 and Table 55 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the TRACE_ROUT SU.

Table 54. TRACE_ROUT SU input data element requirements.

Name	Description	Units	Source
a_s	Propagation angle for start of ray trace	radians	Calling SU
$grdum$	Array of refractivity gradients defined by profile $htdum$ and $refdum$	M-units/ meter	REFINTER SU REFINIT SU
h_s	Height for start of ray trace	meters	Calling SU
$htdum$	Height array for current interpolated profile	meters	REFINTER SU REFINIT SU
ht_{lim}	User-supplied maximum height relative to h_{minter} , i.e., $ht_{lim} = h_{max} - h_{minter}$	meters	TERINIT SU
j_s	Refractive profile index for start of ray trace	N/A	Calling SU
$lvlep$	Number of height/refractivity levels in profile $htdum$, $refdum$	N/A	REFINIT SU
n_{rout}	Number of output height points desired	N/A	Calling CSCI
$rngout$	Array containing all desired output ranges	meters	APMINIT CSC
r_s	Range for start of ray trace	meters	Calling SU

Table 55. TRACE_ROUT SU output data element requirements.

Name	Description	Units
<i>harray</i>	Array containing heights of traced ray at every output range	meters
<i>i_{hmx}</i>	Index in <i>harray</i> where traced height has reached <i>ht_{lim}</i>	N/A

5.1.24 Troposcatter Initialization (TROPOINT) SU

The purpose of the TROPOINT SU is to initialize all variables and arrays needed for subsequent troposcatter calculations. The tangent range and tangent angle are determined from the source and the tangent range and tangent angles are determined for all receiver heights and stored in arrays.

Upon entering the SU, the array $\vartheta 1t$ is allocated for size i_{PE} and initialized to 0. Next, the GET_K SU is referenced to determine the effective earth radius factor a_{ek} based on a ray launched at the critical angle traced to ht_{lim} . The array $\vartheta 0$, containing angles used in determining the common volume scattering angle is then determined from

$$\vartheta 0_i = \frac{rngout_i}{a_{ek}}; \quad \text{for } i = 1, 2, 3, \dots, n_{rout}.$$

A constant needed in the troposcatter calculation, r_f , is determined from 0.0419 times the frequency f_{MHz} . A second constant needed in the troposcatter calculation, rt_1 , is found from r_f times the adjusted transmitting antenna height ant_{ref} . Next, the tangent angle from the source, $\vartheta 1_s$, for smooth surface is computed from

$$\vartheta 1_s = \frac{\sqrt{twoka * ant_{ref}}}{a_{ek}}$$

The variable α_d is determined from

$$\begin{aligned} \alpha_{ld} &= 20 \text{LOG}_{10}(f(\alpha_d)) \\ \alpha_d &= \vartheta 1_s + 10^{-6} \end{aligned},$$

where α_d represents the lowest direct ray angle in the RO region, and $f(\alpha_d)$ is the antenna pattern factor, obtained from referencing the ANTPAT SU, for the direct angle.

The minimum range, r_{hor1} , at which the diffraction field solutions are applicable and the intermediate region ends is determined for smooth surface and zero receiver height. The variable r_{hor1} is given by

$$r_{hor1} = \sqrt{twoka * ant_{ref}} + 230200.0 \left(\frac{e_k^2}{f_{MHz}} \right)^{.3333},$$

where e_k is the effective Earth's radius factor value obtained from the GET_K SU.

Next, the tangent ranges and angles for all output receiver heights are computed and stored in arrays $d2s$ and $\vartheta2s$, respectively. The minimum ranges at which diffraction field solutions are applicable (for smooth surface) for all output receiver heights are determined and stored in array rdt . Height differences between ant_{ref} and each output receiver height are also computed and stored in $adif$. These arrays are given by

$$\begin{aligned} d2s_i &= \sqrt{2a_{ek} zout_i}, \\ \vartheta2s_i &= -\frac{d2s_i}{a_{ek}}, \quad i = 0, 1, 2, \dots, n_{zout} \\ rdt_i &= r_{hor1} + d2s_i \\ adif_i &= ant_{ref} - zout_i, \end{aligned}$$

where the computation is performed for each i^{th} output receiver height $zout_i$, provided $zout_i$ is greater than or equal to 0, and i ranges from 1 to n_{zout} .

If f_{ter} is '.true.', then the tangent angles $\vartheta1t$ from the source at every PE range is determined as

$$\vartheta1t = \frac{ant_{ref} - tyh_j}{j\Delta r_{PE}} + \frac{j\Delta r_{PE}}{2a_{ek}}, \quad j = 1, 2, 3, \dots, i_{PE}.$$

The index counter j_{Ω} (used in the TROPOSCAT SU) is initialized 1. Finally, the troposcatter loss term $tlst_{wr}$, used in the TROPOSCAT SU is given by

$$tlst_{wr} = 54.9 + 30 \mathbf{LOG}_{10}(f_{MHz}) - \alpha_{ld}.$$

Table 56 and Table 57 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the TROPOINT SU.

Table 56. TROPOINT SU input data element requirements.

Name	Description	Units	Source
ant_{ref}	Transmitting antenna height relative to h_{minter}	meters	TERINIT SU
f_{MHz}	Frequency	MHz	Calling CSCI
f_{ter}	Logical flag indicating if terrain profile has been specified: '.true.' = terrain profile specified '.false.' = terrain profile not specified	N/A	TERINIT SU
i_{PE}	Number of PE range steps	N/A	PEINIT SU
n_{rout}	Integer number of output range points desired	N/A	Calling CSCI
n_{zout}	Integer number of output height points desired	N/A	Calling CSCI
$rngout$	Array containing all desired output ranges	meters	APMINIT CSC
tyh	Adjusted height points of terrain profile at every PE range step	meters	PEINIT SU
$zout$	Array containing all desired output heights referenced to h_{minter}	meters	APMINIT CSC

Table 57. TROPOINT SU output data element requirements.

Name	Description	Units
a_{ek}	Effective earth radius	meters
$adif$	Height differences between ant_{ref} and all output receiver heights	meters
$d2s$	Array of tangent ranges for all output receiver heights over smooth surface	meters
e_k	Effective earth radius factor	N/A
$jt2$	Index counter for tx and ty arrays indicating location of receiver range	N/A
rdt	Array of minimum ranges at which diffraction field solutions are applicable (for smooth surface) for all output receiver heights.	meters
rf	Constant used for troposcatter calculations	meters ⁻¹
rt_1	rf multiplied by ant_{ref}	N/A
ϑ_0	Array of angles used to determine common volume scattering angle	radians
ϑ_{1s}	Tangent angle from source (for smooth surface)	radians
ϑ_{1t}	Array of tangent angles from source height - used with terrain profile	radians
ϑ_{2s}	Array of tangent angles from all output receiver heights - used with smooth surface	radians
$tlst_{wr}$	Troposcatter loss term used in the TROPOSCAT SU	dB

5.1.25 Starter Field Initialization (XYINIT) SU

The purpose of the XYINIT SU is to calculate the complex PE solution at range zero.

Upon entering this SU, several constant terms which will be employed over the entire PE mesh are calculated. The PE mesh is defined by the number of points in the mesh, n_{fft} , and by the mesh size Δp . The constant terms include : (1) the angle difference between mesh points in p-space $\Delta\theta$; (2) a height-gain value at the source (transmitter) $antk_o$; and (3) the normalization factor s_{gain} used in the determination of the complex array containing the field U . The normalization factor s_{gain} is given by

$$s_{gain} = \frac{\sqrt{\lambda}}{z_{max}} .$$

The height-gain value $antk_o$ at the source (transmitter) is given by

$$antk_o = k_o ant_{ht}$$

where ant_{ht} is the transmitting antenna height above the local ground in meters.

The complex PE solution U is determined from the antenna pattern factors, elevation angle, and normalization factor according to

$$\begin{aligned} U_j &= c_a s_{gain} \left[f(\alpha_d) e^{-ip_j antk_o} - f(-\alpha_d) e^{ip_j antk_o} \right]; \text{ H pol} \\ U_j &= c_a s_{gain} \left[f(\alpha_d) e^{-ip_j antk_o} + f(-\alpha_d) e^{ip_j antk_o} \right]; \text{ V pol} \\ \alpha_d &= \text{SIN}^{-1}(p_j), \\ c_a &= (1 - p_j^2)^{-3/4} \end{aligned}$$

where $p_j = j\Delta\theta$ and the antenna pattern factors $f(\alpha_d)$ for the direct path and $f(-\alpha_d)$ for the reflected path are determined by referencing the ANTPAT SU. The index j varies from 0 to n_{fft} .

Next, the upper $\frac{1}{4}$ of the field is filtered. A cosine-tapered (Tukey) filter array $filt$ is used for this purpose. The filtered PE field U is given by

$$U_j = filt_{j-n_{34}} U_j; \quad j = n_{34}, n_{34} + 1, n_{34} + 2, \dots, n_{fft} .$$

Finally, the DRST SU is referenced for both the real and imaginary components to transform the complex PE field to z-space before exiting the SU.

Table 58 and Table 59 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the XYINIT SU.

Table 58. XYINIT SU input data element requirements.

Name	Description	Units	Source
ant_{ht}	Transmitting antenna height above local ground	meters	Calling CSCI
$\Delta\theta$	Angle bin width (i.e., incremental sine(theta))	radians	PEINIT SU
$filt$	Cosine-tapered (Tukey) filter array	N/A	PEINIT SU
i_{pol}	Polarization flag: 0 = horizontal polarization 1 = vertical polarization	N/A	Calling SU
k_o	Free-space wave number	meters ⁻¹	APMINIT CSC
λ	Wavelength	meters	APMINIT CSC
ln_{fft}	Power of 2 transform size, i.e., $n_{fft}=2^{ln_{fft}}$	N/A	FFTPAR SU
n_{fft}	Transform size	N/A	FFTPAR SU
n_{34}	$\frac{3}{4} n_{fft}$	N/A	APMINIT CSC
z_{max}	Total height of the FFT/PE calculation domain	meters	FFTPAR SU

Table 59. XYINIT SU output data element requirements.

Name	Description	Units
U	Transform of complex field	$\mu\text{V/m}$

5.2 ADVANCED PROPAGATION MODEL STEP (APMSTEP) CSC

The purpose of the APMSTEP SU is to advance the entire APM CSCI algorithm one output range step, referencing various SUs to calculate the propagation loss at the current output range.

Upon entering the APMSTEP SU, the current output range r_{out} is updated, the gaseous absorption loss, gas_{loss} , in dB and all $mpfl$ array integer indices for the various calculation regions are initialized. The PESTEP SU is then referenced to determine all propagation loss values within the PE calculation region. If the PE-only option is specified ($PE_{flag} = \text{'true.'}$) then $mpfl$ is returned with integer indices j_{ps} and j_{pe} , corresponding to the start and end, respectively, of propagation factor and loss values within $mpfl$. Otherwise, the SU proceeds with the following steps 1 through 3.

1. If APM is executing using the airborne model ($i_{hybrid} = 0$) then the starting index j_{as} for the airborne region is initialized to the maximum of 0 and i_{zg} plus i_o . The ending

index within this region j_{ae} is determined by performing an iterative search to find the index at which the first occurrence of $zout_j$ is greater than $htfe_{i_{stp}}$. j_{ae} is then set equal to the index j . The AIRBORNE SU is then referenced to compute the loss for the lower FE region, provided j_{as} is less than j_{ae} . Upon returning, j_{start} is then set equal to j_{as} , j_{as} is set equal to $j_{pe}+1$, j_{ae} is set equal to n_{zout} , and the AIRBORNE SU is again referenced to compute loss for the upper FE region. The SU then proceeds with step 3.

2. If APM is executing under the full hybrid mode ($i_{hybrid} = 1$) and the current output range is less than the range at which the XO region begins ($r_{out} < r_{atz}$), the following steps 2.a and 2.b are performed.
 - a. The starting and ending *mpfl* array indices for FE calculations, j_{fs} and j_{fe} , respectively, are determined. For ranges less than 2.5 km, j_{fs} is set equal to the maximum of 0 and i_{zg} plus i_o , and j_{fe} is set equal to n_{zout} . For ranges greater than 2.5 km, j_{fs} is set equal to the maximum of $j_{pe}+1$, or $j+1$, where j is the first occurrence of $zout_j$ that is greater than $htfe_{i_{stp}}$ (the output height index which corresponds to the height just above the FE 5° angle limit). The ending index j_{fe} is set equal to n_{zout} . The FEM SU is then referenced and propagation factor and loss values within the FE region are computed and returned in *mpfl*.
 - b. If the current output range is greater than 2.5 km, then the starting and ending *mpfl* array indices for RO calculations, j_{rs} and j_{re} , respectively, are determined. These indices are based on the values of j_{ps} , j_{pe} , j_{fs} , and j_{fe} such that at every range step, j_{rs} will always be greater than the ending index of the PE region (j_{pe}) and j_{re} will be less than the starting index of the FE region (j_{fs}). The ROLOSS SU is then referenced, and propagation factor and loss values within the RO region are computed and returned in *mpfl*.
3. Once the various propagation factor and loss within the various regions have been calculated, the ending index j_{end} of valid values within *mpfl* is given by the maximum of j_{pe} , j_{fe} , j_{re} , and j_{ae} .

Upon exiting, if the final output range step has been reached, the integer counter j_{t2} , associated with troposcatter calculations, is initialized to 1.

Table 60 and Table 61 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the APMSTEP SU.

Table 60. APMSTEP CSC input data element requirements.

Name	Description	Units	Source
gas_{att}	Gaseous absorption attenuation rate	dB/km	GASABS SU
$htfe$	Array containing the height at each output range separating the FE region from the RO region (full hybrid mode), or the FE region from the PE region (partial hybrid mode)	meters	FILLHT SU
ht_{lim}	Maximum height relative to h_{minter}	meters	TERINIT SU
i_{hybrid}	Integer indicating which sub-models will be used: 0 = pure PE model 1 = full hybrid model (PE + FE + RO + XO) 2 = partial hybrid model (PE + XO)	N/A	GETMODE SU
i_o	Starting index for $mpfl$ array: 0 = 1 st calculated output point is at surface 1 = 1 st calculated output point is at height Δz_{out}	N/A	Calling CSCI
i_{stp}	Current output range step index	N/A	Calling CSCI
i_{zg}	Number of output height points corresponding to local ground height at current output range r_{out}	N/A	CALCLOS SU
no_{PE}	Integer flag indicating if PE calculations are needed: 0 = PE calculations needed 1 = no PE calculations needed	N/A	GETTHMAX SU
$nrout$	Integer number of output range points desired	N/A	Calling CSCI
$nzout$	Integer number of output height points desired	N/A	Calling CSCI
PE_{flag}	Flag to indicate use of PE algorithm only: 'true.' = only use PE sub-model 'false.' = use automatic hybrid model	N/A	Calling CSCI
r_{atz}	Range at which z_{lim} is reached (used for hybrid model)	meters	APMINIT CSC
$rngout$	Array containing all desired output ranges	meters	APMINIT CSC
r_{pest}	Range at which loss values from the PE model will start being calculated	meters	GETTHMAX SU
$rtst$	Range at which to begin RO calculations (equal to 2.5 km)	meters	APM_MOD
$zout$	Array containing all desired output heights referenced to h_{minter}	meters	APMINIT CSC

Table 61. APMSTEP CSC output data element requirements.

Name	Description	Units
j_{end}	Index at which valid loss values in $mpfl$ end	N/A
j_{start}	Index at which valid loss values in $mpfl$ start	N/A
gas_{loss}	Gaseous absorption loss at range r_{out}	dB
$jt2$	Index counter for tx and ty arrays indicating location of receiver range	N/A
$mpfl$	Propagation factor and loss array	cB
r_{out}	Current desired output range	meters

5.2.1 Airborne Hybrid Model (AIRBORNE) SU

The purpose of the AIRBORNE SU is to determine propagation factor and loss based on flat-earth calculations for the direct ray path only for regions above and below the PE maximum propagation angle.

Upon entering the SU, the square of the current range, r_{sq} , is initialized to $rsqrd_{i_{stp}}$. Next, the earth curvature height correction factor r_{sqk} is initialized according to the calculation region:

$$r_{sqk} = \frac{r_{sq}}{twoka}; \quad \text{above PE region}$$

$$r_{sqk} = \frac{r_{sq}}{twoka_{down}}; \quad \text{below PE region.}$$

The following steps 1 through 2 are performed for all heights within array $zoutma$ with index j varying from j_{as} to j_{ae} .

1. First, the direct ray angle is computed as

$$\alpha = \text{TAN}^{-1} \left(\frac{zoutma_j - r_{sqk}}{r_{out}} \right).$$

2. Next, if the direct ray α is greater than the tangent angle α_{ter} produced from the antenna height to the current terrain height, or the calculations are for the upper PE region, then steps 2.a through 2.d are performed.
 - a. The ANTPAT SU is referenced to obtain the antenna pattern factor $f(\alpha)$ for the direct ray.

- b. The path length of the direct ray is then computed:

$$r_1 = \sqrt{(z_{outma_j} - r_{sqk})^2 + r_{sq}} .$$

- c. The propagation factor (F_{dB}) and loss (L) are then computed from

$$L = 20 \text{LOG}_{10}(r_1) + pl_{cnst} - 20 \text{LOG}_{10}(\text{MAX}(f(\alpha), 10^{-13})) + r_1 gas_{att}$$

$$F_{dB} = 20 \text{LOG}_{10}(r_1) + pl_{cnst} - L .$$

Note that F_{dB} above is actually 20 times the logarithm of the propagation factor F as defined in most text books.

- d. Lastly, L and F_{dB} are multiplied by 10 and rounded to the nearest integer then stored in array *mpfl*. Once the loss has been computed for all heights, the SU is then exited.

Table 62 and Table 63 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the AIRBORNE SU.

Table 62. AIRBORNE SU input data element requirements.

Name	Description	Units	Source
α_{ter}	Tangent angle from antenna height to terrain height at current range	radians	CALCLOS SU
gas_{att}	Gaseous absorption attenuation rate	dB/km	GASABS SU
i_{flag}	Flag indicating which portion of the FE region is being computed. 0 = loss is computed for heights above PE region 1 = loss is computed for heights below PE region	N/A	Calling SU
i_{stp}	Current output range step index	N/A	Calling CSCI
j_{ae}	Ending index within <i>mpfl</i> of FE loss values	N/A	Calling SU
j_{as}	Starting index within <i>mpfl</i> of FE loss values	N/A	Calling SU
r_{out}	Current output range	meters	Calling SU
pl_{cnst}	Constant used in determining propagation loss ($pl_{cnst} = 20 \log_{10}(2 k_o)$)	dB/m	APMINIT CSC
$rsqrd$	Array containing the square of all desired output ranges	meters ²	APMINIT CSC
$twoka$	Twice the effective earth radius	meters	GETTHMAX SU
$twoka_{down}$	Twice the effective earth radius for downward path	meters	GETTHMAX SU
$zoutma$	Array output heights relative to "real" ant_{ref}	meters	APMINIT CSC

Table 63. AIRBORNE SU output data element requirements.

Name	Description	Units
<i>mpfl</i>	Array of propagation factor and loss	μV/m

5.2.2 Calculate Propagation Loss (CALCLOS SU)

The purpose of the CALCLOS SU is to determine the propagation factor and loss at each output height point at the current output range.

Upon entering the SU, several variables are initialized. The output range, r_{out} , is updated based on the current range step i_{stp} . The height of the terrain at the current and last ranges, y_{ch} and y_{lh} , respectively, are determined relative to the reference height, hm_{ref} .

Next, the interpolated ground height, z_{int} , at the current output range and the number of vertical output points, i_{zg} , that correspond to this ground height are determined. First, the interpolated ground height is given by

$$z_{int} = y_{last} + (y_{cur} - y_{last})xx$$

where the parameter xx is given in terms of the PE range step Δr_{PE} by

$$xx = \frac{r_{out} - r_{last}}{\Delta r_{PE}} .$$

Having determined z_{int} , i_{zg} is then computed from

$$i_{zg} = \text{INT} \left(\frac{z_{int} - hm_{ref}}{\Delta z_{out}} \right) .$$

where Δz_{out} is the output height increment. Next, all elements in array *mpfl* from 1 to i_{zg} are set to zero, and the index j_{start} , representing beginning valid loss values in the *mpfl* array, is set to the maximum of 0 or i_{zg} , plus i_o .

If the airborne model is required ($i_{hybrid}=0$), then the maximum tangent angle α_{ter} from the antenna height to the terrain height at the current range is determined from

$$\alpha_{ter} = \text{MAX} \left[\text{TAN}^{-1} \left(\frac{z_{int} - ant_{ref} - \frac{r_{out}^2}{twoka_{down}}}{r_{out}} \right), \alpha_{ter} \right],$$

where α_{ter} in the argument above is the maximum angle computed from previous references to the CALCLOS SU.

If the current output range is greater than the range r_{pest} at which PE solutions are valid, then the calculation of loss values is begun. If this condition is not satisfied, then the *mpfl* array is set to -1000 for values of the array index from j_{start} up to and including the number of output height points desired (n_{zout}), j_{end} is set equal to j_{start} and the SU is exited.

Once it is determined that loss calculations will be performed, several parameters are computed. Both parameters i_{p1} and i_{p2} are first set to 0. If the logical variable f_{ter} is '.true.', then a terrain case is being performed. The two indices i_{p1} and i_{p2} are given by

$$i_{p1} = \text{MAX} \left(0, \text{INT} \left\{ \frac{y_{lh}}{\Delta z_{out}} \right\} \right)$$

$$i_{p2} = \text{MAX} \left(0, \text{INT} \left\{ \frac{y_{ch}}{\Delta z_{out}} \right\} \right).$$

These indices indicate the first output height point in array *zout* where propagation loss will be computed at the last and current PE ranges. Next, the output heights $zout_{i_{p1}}$ and $zout_{i_{p2}}$, relative to y_{last} and y_{cur} , respectively, are checked to make sure they are positive. If not, the two indices i_{p1} and i_{p2} are incremented by a value of 1. For values of the array index from 0 up to i_{p1} , the array of propagation factors *rfac1* at valid height points for range r_{last} are set to 0. For values of the array index from 0 up to i_{p2} , the array of propagation factors *rfac2* at valid height points for range r_{out} are also set 0.

If j_{start} is less than both i_{p1} and i_{p2} then the variable i_{zg} is recomputed as

$$i_{zg} = \text{MAX}(i_{zg}, \text{MIN}(i_{p1}, i_{p2}))$$

and the *mpfl* array at index i_{zg} is set equal to -999. j_{start} is then recomputed as the minimum of 0 and i_{zg} , plus i_o .

Next, the height/integer value, j_{end} , indicating the end of valid loss values, is determined as

$$j_{end} = \text{MAX} \left[0, \text{INT} \left(\frac{z_{lim} - hm_{ref}}{\Delta z_{out}} \right) \right] \quad \text{if } PE_{flag} \text{ is 'true.', otherwise}$$

$$j_{end} = \text{MAX} \left(0, \text{NINT} \left\{ \frac{\text{MIN} [z_{lim}, \text{MAX} (z_{int}, hlim_{i_{stp}})] - hm_{ref}}{\Delta z_{out}} \right\} \right).$$

where i_{stp} is the current output range step, and $hlim_{i_{stp}}$ is the height at the current output range step separating the PE region from the FE, RO, or XO regions. Finally, j_{end} is given by the minimum of j_{end} (as computed above) and n_{zout} . If j_{end} is less than j_{start} , then the $mpfl$ array for elements $j_{end}+1$ to n_{zout} is set equal to -1000 and the SU is exited.

The propagation loss values are determined from the propagation factors $rfac1_i$ and $rfac2_i$ and from the parameter xx defined earlier in this section. If r_{loglst} ($10 \text{ LOG}(r_{last})$) is greater than zero (it is initialized to 0 for $i_{stp}=1$), then the GETPFAC SU is referenced to determine the propagation factor $rfac1_i$, which is given by

$$rfac1_i = \text{GETPFAC}(U_{last}, r_{loglst}, \Delta z_{PE}, z_{out_i} - y_{last}), \quad i = i_{p1}, i_{p1} + 1, \dots, j_{end}.$$

where U_{last} is the complex field array at the previous PE range. Next, the propagation factor $rfac2_i$ is given by

$$rfac2_i = \text{GETPFAC}(U, r_{log}, \Delta z_{PE}, z_{out_i} - y_{cur}), \quad i = i_{p2}, i_{p2} + 1, \dots, j_{end},$$

where U is the complex field array at the current PE range, and r_{log} is $10 \text{ LOG}(r)$.

Next, if using the partial hybrid mode (PE & XO models), heights corresponding to areas outside the valid PE calculation region are determined and propagation loss is set equal to -1000 within $mpfl$ for those heights. If using the full or partial hybrid modes, the propagation factor at the last PE height point is determined at both the previous and current PE ranges. Linear interpolation is then performed to compute the propagation loss at range r_{out} and height z_{lim} . The loss and height are then stored in array $ffrout$ for subsequent interpolation in the EXTO SU.

Next, the propagation factor and loss at range r_{out} is found by interpolating between the current and previous PE ranges. The propagation factor in dB, and the propagation loss at range r_{out} is given by

$$F_{dB} = rfac1_i + (rfac2_i - rfac1_i)xx$$

$$rloss_i = fsl_{i_{stp}} - F_{dB} \quad ; \quad i = j_{start}, j_{start}+1, \dots, j_{end}$$

where $fsl_{i_{stp}}$ is the free-space loss in dB at range r_{out} .

If the troposcatter calculation flag T_{ropo} is ‘.true.’, then the TROPOSCAT SU is referenced to compute troposcatter loss from height $zout_{j_{start}}$ to $zout_{j_{end}}$ and this is added, if necessary, to $rloss$.

Finally, the gaseous absorption loss gas_{loss} is added to $rloss$ and the loss and propagation factor in centibels is given by

$$mpfl_{1,i} = \text{NINT}(10 * rloss_i)$$

$$mpfl_{2,i} = \text{NINT}\left(10 * \left(fsl_{i_{stp}} - rloss_i\right)\right); \quad i=j_{start}, j_{start}+1, \dots, j_{end}$$

with the remaining elements in $mpfl$ set equal to -1000 (i.e., $mpfl_k = -1000$ for $k = j_{end}+1$ to n_{zout}) before exiting.

Table 64 and Table 65 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the CALCLOS SU.

Table 64. CALCLOS SU input data element requirements.

Name	Description	Units	Source
ant_{ref}	Transmitting antenna height relative to the reference height h_{minter}	meters	TERINIT SU
Δr_{PE}	PE range step	meters	APMINIT CSC
Δz_{out}	Output height increment	meters	APMINIT CSC
Δz_{PE}	PE mesh height increment (bin width in z-space)	meters	FFTPAR SU
fsl	Free space loss array for output ranges	dB	APMINIT CSC
f_{ter}	Logical flag indicating if terrain profile has been specified: .true. = terrain profile specified .false. = terrain profile not specified	N/A	TERINIT SU
gas_{loss}	Gaseous absorption loss at range r_{out}	dB	APMSTEP CSC
$hlim$	Array containing height at each output range separating the RO region from the PE (at close ranges) and XO (at far ranges) regions	meters	GETTHMAX SU
hm_{ref}	Height relative to h_{minter}	meters	TERINIT SU
$htfe$	Array containing the height at each output range separating the FE region from the RO region (full hybrid mode), or the FE region from the PE region (partial hybrid mode)	meters	FILLHT SU
i_{hmx}	Index in $harray$ where traced height has reached ht_{lim}	N/A	TRACE_ROUT SU

Table 64. CALCLOS SU input data element requirements. (Continued)

Name	Description	Units	Source
i_{hybrid}	Integer indicating which sub-models will be used: 0 = pure PE model 1 = full hybrid model (PE + FE + RO + XO) 2 = partial hybrid model (PE + XO)	N/A	APMINIT CSC
i_o	Starting index for $mpfl$ array: 0 = 1 st calculated output point is at surface 1 = 1 st calculated output point is at height Δz_{out}	N/A	APMINIT CSC
i_{stp}	Current output range step index	N/A	Calling SU
i_{xo}	Number of range steps in XO calculation region	N/A	APMINIT CSC
n_{zout}	Integer number of output height points desired	N/A	Calling CSCI
PE_{flag}	Flag to indicate use of PE algorithm only: 'true.' = only use PE sub-model 'false.' = use automatic hybrid model	N/A	Calling CSCI
r_{atz}	Range at which z_{lim} is reached (used for hybrid model)	meters	APMINIT CSC
r_{last}	Previous PE range	meters	Calling SU
r_{log}	10 log(PE range r)	N/A	PESTEP SU
r_{loglst}	10 log(previous PE range r_{last})	N/A	PESTEP SU
$rngout$	Array containing all desired output ranges	meters	APMINIT CSC
r_{pest}	Range at which PE loss values will start being calculated	meters	GETTHMAX SU
T_{ropo}	Troposcatter calculation flag: 'false.' = no troposcatter calcs 'true.' = troposcatter calcs	N/A	Calling CSCI
$twoka_{down}$	Twice the effective earth radius for downward path	meters	GET_K SU
U	Complex field at current PE range r	$\mu V/m$	PESTEP SU
U_{last}	Complex field at previous PE range r_{last}	$\mu V/m$	PESTEP SU
y_{cur}	Height of ground at current range r	meters	PESTEP SU
y_{last}	Height of ground at previous range r_{last}	meters	PESTEP SU
$zlim$	Height limit for PE calculation region	meters	GETTHMAX SU
$zout$	Array containing all desired output heights referenced to h_{minter}	meters	APMINIT CSC

Table 65. CALCLOS SU output data element requirements.

Name	Description	Units
α_{ter}	Tangent angle from antenna height to terrain height at current range	radians
$ffrout$	Array of propagation factors at each output range beyond r_{atz} and at height z_{lim}	dB
i_{zg}	Number of output height points corresponding to local ground height at current output range r_{out}	N/A
j_{end}	Index at which valid loss values in $mpfl$ end	N/A
j_{start}	Index at which valid loss values in $mpfl$ begin	N/A
$mpfl$	2-dimensional propagation factor and loss array	cB
$rfac1$	Propagation factor at valid output height points from PE field at range r_{last}	dB
$rfac2$	Propagation factor at valid output height points from PE field at range r	dB
$rloss$	Propagation loss	dB

5.2.3 DOSHIFT SU

The purpose of the DOSHIFT SU is to shift the field by the number of bins, or PE mesh heights corresponding to the local ground height.

Upon entry, the number of bins to be shifted is determined. First, the difference y_{diff} between the height of the ground y_{last} at the previous range and that at the current PE range y_{cur} is determined from

$$y_{diff} = y_{cur} - y_{last} .$$

The number of bins to be shifted, k_{bin} , is found from

$$k_{bin} = \text{NINT} \left(\frac{|y_{diff}|}{\Delta z_{PE}} \right) .$$

The PE solution U is then shifted downward if the local ground is currently at a positive slope ($y_{diff} > 0$), upward if the local ground is at a negative slope ($y_{diff} < 0$), and otherwise not shifted. When the PE solution has been shifted down, the value of the PE solution U for the upper k_{bin} elements are set to zero. Likewise, when the PE solution has been shifted upwards, the lower k_{bin} elements are set to zero.

Table 66 and Table 67 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the DOSHIFT SU.

Table 66. DOSHIFT SU input data element requirements.

Name	Description	Units	Source
Δz_{PE}	PE mesh height increment (bin width in z-space)	meters	FFTPAR SU
n_{fft}	Transform size	N/A	FFTPAR SU
n_{m1}	$n_{fft} - 1$	N/A	PEINIT SU
U	Complex field at range r	$\mu\text{V/m}$	PESTEP SU
y_{cur}	Height of ground at current range r	meters	PESTEP SU
y_{last}	Height of ground at previous range r_{last}	meters	PESTEP SU

Table 67. DOSHIFT SU output data element requirements.

Name	Description	Units
U	Complex field at range r	$\mu\text{V/m}$

5.2.4 Discrete Sine/Cosine Fast-Fourier Transform (DRST) SU

A function with a common period, such as a solution to the wave equation, may be represented by a series consisting of sines and cosines. This representation is known as a Fourier series. An analytical transformation of the function, known as a Fourier transform, may be used to obtain a solution for the function.

The solution to the PE approximation to Maxwell's wave equation is obtained by using such a Fourier transformation function. The APM CSCI uses only the real-valued sine or cosine transformation in which the real and imaginary parts of the PE equation are transformed separately. Which transform is performed is dependent on the value of an integer flag provided by the calling SU. The Fourier transformation provided with the APM CSCI is described by Bergland (Ref. 1) and Cooley, Lewis, and Welsh (Ref. 2). The FORTRAN source code is listed in Appendix A.

Other sine/cosine fast-Fourier transform (FFT) routines are available in the commercial market, and such a sine/cosine FFT may already be available within another NITES-NC CSCI. The selection of which FFT ultimately used by the APM CSCI is left to the application designer as every sine/cosine FFT will have hardware and/or software performance impacts. For this reason, it is beyond the scope of this document to describe the numerical implementation of the FFT algorithm.

Table 68 and Table 69 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the DRST SU.

Table 68. DRST input data element requirements.

Name	Description	Units	Source
i_{flag}	Flag to indicate which transform to perform 0 = cosine transform 1 = sine transform -1 = deallocates all allocated arrays	N/A	Calling SU
ln_{fft}	Power of 2 transform size, i.e., $n_{fft}=2^{ln_{fft}}$	N/A	FFTPAR SU
x	Field array to be transformed - dimensioned $2^{n_{fft}}$ in calling SU	$\mu\text{V/m}$	Calling SU

Table 69. DRST output data element requirements.

Name	Description	Units
x	Transform of field	$\mu\text{V/m}$

5.2.5 Flat-earth Model (FEM) SU

The purpose of the FEM SU is to compute propagation loss at a specified range based upon flat-earth approximations. The following steps 1 through 10, are performed for each APM height output point j from j_{fs} to j_{fe} .

1. The receiver height at the j^{th} output point, z_{outj} , is first adjusted relative to the antenna height for both the direct and reflected ray paths and is also corrected for earth curvature and average refraction. The receiver heights, z_m and z_p , relative to both the real (direct) and image (reflected) antenna height, respectively, are defined as follows:

$$z_m = z_{outma_j} - \frac{r_{out}^2}{twoka}$$

$$z_p = z_{outpa_j} - \frac{r_{out}^2}{twoka}$$

where z_{outma_j} and z_{outpa_j} represent the output height z_{outj} relative to the “real” and “image” antenna heights, respectively, with respect to mean sea level. $twoka$ is twice the effective earth radius as calculated in the FILLHT SU.

2. Next, the point or range, of reflection, $x_{reflect}$, is given by

$$x_{reflect} = r_{out} \frac{ant_{ref}}{z_p}$$

This quantity is used when referencing the GETREFCOEF SU.

3. The elevation angles for the direct- and reflected-path rays, α_d and α_r , respectively, are given as

$$\alpha_d = \text{TAN}^{-1} \left(\frac{z_m}{r_{out}} \right)$$

$$\alpha_r = \text{TAN}^{-1} \left(\frac{z_p}{r_{out}} \right)$$

4. The ANTPAT SU is referenced with the direct-path elevation angle to obtain the antenna pattern factor for the direct-path ray, $f(\alpha_d)$; and with the grazing angle (opposite of the reflected-path ray angle) to obtain the antenna pattern factor for the surface-reflected ray, $f(-\alpha_r)$.
5. The path lengths for both the direct-path, r_1 , and surface-reflected path, r_2 , are computed from simple right triangle calculations, as

$$r_1 = \sqrt{z_m^2 + r_{out}^2},$$

$$r_2 = \sqrt{z_p^2 + r_{out}^2}.$$

6. The GETREFCOEF SU is referenced with the reflected-path ray angle to obtain the amplitude, R_{mag} , and phase angle, φ , of the surface-reflection coefficient.
7. From the two path lengths, the surface-reflection phase lag angle, and the free-space wave number, k_o , the total phase angle is determined as

$$\Omega = (r_2 - r_1)k_o + \varphi.$$

8. The square of the coherent sum of both the direct-path ray and surface-reflected path ray is computed as

$$f_{sum}^2 = \left| f(\alpha_d)^2 + R_{mag}^2 f(-\alpha_r)^2 + 2f(\alpha_d)R_{mag}f(-\alpha_r)\text{COS}(\Omega) \right|.$$

9. The propagation factor in decibels, F_{dB} , is computed as

$$F_{dB} = 10 \text{LOG}_{10} [\text{MAX}(f_{sum}^2, 10^{-25})].$$

A limit of -250 dB was put on F_{dB} to avoid underflow problems.

10. Finally, the propagation factor and loss for the output point $zout_j$ is calculated and rounded to the nearest centibel as

$$mpfl_{1,j} = \text{NINT}\left(10\left(L_{fs} - F_{dB} + r_1 gas_{att}\right)\right)$$

$$mpfl_{2,j} = \text{NINT}\left(10\left(L_{fs} - mpfl_{1,j}\right)\right)$$

where L_{fs} is the free-space loss term in decibels and is given by

$$L_{fs} = 20 \text{LOG}_{10}(r_1) + pl_{cnst}.$$

Table 70 and Table 71 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the FEM SU.

Table 70. FEM SU input data element requirements.

Name	Description	Units	Source
ant_{ref}	Transmitting antenna height relative to h_{minter}	meters	TERINIT SU
gas_{att}	Gaseous absorption	dB/m	GASABS SU
ht_{lim}	Maximum height relative to h_{minter}	meters	TERINIT SU
i_{stp}	Current output range step index	N/A	Calling SU
j_{fe}	Ending index within $mloss$ of FE loss values	N/A	Calling SU
j_{fs}	Starting index within $mloss$ of FE loss values	N/A	Calling SU
k_o	Free-space wave number	meters ⁻¹	APMINIT CSC
pl_{cnst}	Constant used in determining propagation loss ($pl_{cnst} = 20 \log_{10}(2 k_o)$)	dB/m ²	APMINIT CSC
r_{out}	Current output range	meters	Calling SU
$rsqrd$	Array containing the square of all desired output ranges	meters ²	APMINIT CSC
$twoka$	Twice the effective earth's radius	meters	FILLHT SU
$zoutma$	Array output heights relative to "real" ant_{ref}	meters	APMINIT CSC
$zoutpa$	Array output heights relative to "image" ant_{ref}	meters	APMINIT CSC

Table 71. FEM SU output data element requirements.

Name	Description	Units
α_d	Direct path ray angle	radians
$mpfl$	Propagation factor and loss array	cB

5.2.6 Fast-Fourier Transform (FFT) SU

The purpose of the FFT SU is to separate the real and imaginary components of the complex PE field into two real arrays and then reference the DRST SU to transform each portion of the PE solution.

For a transform size, n_{fft} , the real and imaginary parts of the complex PE field array U , respectively, are found for the index i from 0 to n_{fft} :

$$\begin{aligned} xdum_i &= \text{REAL} (U_i) \\ ydum_i &= \text{IMAG} (U_i) \end{aligned}$$

The DRST SU is referenced in turn for $xdum$ and $ydum$ along with ln_{fft} , the power of the transform size to the base 2. The real and imaginary parts of the resulting transform arrays are then converted to the complex array U for i equal 0 to n_{fft} by

$$U_i = \text{CMPLX} (xdum_i, ydum_i) .$$

Table 72 and Table 73 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the FFT SU.

Table 72. FFT SU input data element requirements.

Name	Description	Units	Source
ln_{fft}	Power of 2 transform size, i.e., $n_{fft}=2^{ln_{fft}}$	N/A	FFTPAR SU
n_{fft}	Transform size	N/A	FFTPAR SU
U	Complex field to be transformed	$\mu\text{V/m}$	Calling SU

Table 73. FFT SU output data element requirements.

Name	Description	Units
U	Transform of complex field	$\mu\text{V/m}$

5.2.7 Free-Space Range Step (FRSTP) SU

The purpose of the FRSTP SU is to propagate the complex PE solution in free space by one range step.

Upon entry the PE field, $farray$, is transformed to p-space (Fourier space) and its array elements are multiplied by corresponding elements in the free-space propagator array, $frsp$. Before exiting, the PE field is transformed back to z-space. Both transforms are performed by referencing the FFT SU.

Table 74 and Table 75 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the FRSTP SU.

Table 74. FRSTP SU input data element requirements.

Name	Description	Units	Source
<i>farray</i>	Field array to be propagated one range step in free space	$\mu\text{V/m}$	Calling SU
<i>frsp</i>	Complex free-space propagator term array	N/A	PEINIT SU
<i>n_{m1}</i>	<i>n_{fft}</i> - 1	N/A	PEINIT SU

Table 75. FRSTP SU output data element requirements.

Name	Description	Units
<i>farray</i>	Propagated field array	$\mu\text{V/m}$

5.2.8 FZLIM SU

The purpose of the FZLIM SU is to calculate and store the outward propagation angle and propagation factor at the top of the PE region for the current PE range. The following steps 1 through 5 are performed for each reference to the FZLIM SU.

1. The GETPFAC SU is referenced to determine the propagation factor F_{dB} at height $z_{lim} - y_{cur}$.
2. If this is the first reference to the FZLIM SU ($iz = 1$), then the GETPFAC SU is referenced to determine the propagation factor, F_{dBst} , at the previous PE range. A linear interpolation is performed on F_{dB} and F_{dBst} to compute the propagation factor at range r_{atz} where the XO region begins. The interpolated propagation factor and the outward propagation angle, Fr_{atz} and a_{atz} , respectively, are stored in array *ffacz*. Next, a reference to the SAVEPRO SU is made to store the refractivity profile at the current range from height z_{lim} to the maximum desired output height.
3. A reference is made to the SPECEST SU to determine the outward propagation angle, ϑ_{out} . The counter *iz* is incremented, but is limited to iz_{max} . The propagation factor F_{dB} , current PE range *r*, and ϑ_{out} (with maximum limit of a_{atz}) are stored in *ffacz_{1,iz}*, *ffacz_{2,iz}*, and *ffacz_{3,iz}*, respectively.
4. If *iz* is greater than 2, then the propagation angle is checked and slightly altered to avoid extreme spiking when using these angles in the XO region. If *f_{ier}* is ‘.false.’, then the angle stored in *ffacz* is the smaller of ϑ_{out} or the previously stored angle.

Now, if f_{ter} is '.false.', or conversely, if f_{ter} is '.true.' AND iz is less than or equal to 10, then the iz^{th} angle stored is adjusted and given by

$$\alpha_{dif} = ffacz_{3,iz} - ffacz_{3,iz-1}$$

$$ffacz_{3,iz} = ffacz_{3,iz-1} \pm \text{MIN}(\alpha_{dif}, 10^{-4})$$

where '+' or '-' is used depending on the sign of α_{dif} .

- Before exiting, a final reference to the SAVEPRO SU is made to store the refractivity profile from height z_{lim} to the maximum desired output height at the current range.

Table 76 and Table 77 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the FZLIM SU.

Table 76. FZLIM SU input data element requirements.

Name	Description	Units	Source
a_{atz}	Local ray or propagation angle at height z_{lim} and range r_{atz}	radians	APMINIT CSC
Δr_{PE}	PE range step	meters	APMINIT CSC
Δz_{PE}	PE mesh height increment (bin width in z-space)	meters	FFTPAR SU
f_{ter}	Logical flag indicating if terrain profile has been specified: .true. = terrain profile specified .false. = terrain profile not specified	N/A	TERINIT SU
iz	Number of propagation factor, range, angle triplets stored in $ffacz$	N/A	APMINIT CSC FZLIM SU
iz_{max}	Maximum number of points allocated for arrays associated with XO calculations	N/A	APMINIT CSC
r	Current PE range	meters	Calling SU
r_{atz}	Range at which z_{lim} is reached (used for hybrid model)	meters	APMINIT CSC
r_{last}	Previous PE range	meters	Calling SU
r_{log}	$10 \log_{10}(\text{PE range } r)$	N/A	PESTEP SU
r_{loglst}	$10 \log_{10}(\text{previous PE range } r_{last})$	N/A	PESTEP SU
U	Complex PE field at range r	$\mu\text{V/m}$	PESTEP SU
U_{last}	Complex PE field at range r_{last}	$\mu\text{V/m}$	PESTEP SU
y_{cur}	Height of ground at current range r	meters	PESTEP SU
y_{last}	Height of ground at previous range r_{last}	meters	PESTEP SU
z_{lim}	Height limit for PE calculation region	meters	GETTHMAX SU

Table 77. FZLIM SU output data element requirements.

Name	Description	Units
<i>ffacz</i>	Array containing propagation factor, range, and propagation angle at z_{lim}	dB, meters, radians
<i>iz</i>	Number of propagation factor, range, angle triplets stored in <i>ffacz</i>	N/A

5.2.9 Get Propagation Factor (GETPFAC) SU

The purpose of the GETPFAC SU is to determine the propagation factor at a specified height.

First, linear interpolation is performed on the magnitudes of the PE field at bins k and $k+1$ to determine the magnitude p_{mag} of the field at the receiver height, z_r :

$$p_{mag} = |U_k| + f_r \left(|U_{k+1}| - |U_k| \right)$$

where the interpolation fraction f_r is determined from

$$f_r = \frac{z_r}{\Delta z_{PE}} - k; \quad k\Delta z_{PE} \leq z_r < (k+1)\Delta z_{PE}$$

$$k = \text{INT} \left(\frac{z_r}{\Delta z_{PE}} \right) .$$

p_{mag} is constrained to be not less than 10^{-12} $\mu\text{V/m}$. Finally, the propagation factor in dB, F_{dB} is given by

$$F_{dB} = 20 \text{LOG}_{10} \left(\text{MAX} \left(p_{mag}, 10^{-12} \right) \right) + r_{log} .$$

Table 78 and Table 79 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the GETPFAC SU.

Table 78. GETPFAC SU input data element requirements.

Name	Description	Units	Source
Δz_{PE}	PE mesh height increment (bin width in z-space)	meters	Calling SU
r_{log}	$10 \log_{10}(\text{PE range } r)$	N/A	Calling SU
U	Complex PE field at range r	$\mu\text{V/m}$	Calling SU
z_r	Receiver height	meters	Calling SU

Table 79. GETPFAC SU output data element requirements.

Name	Description	Units
F_{dB}	Propagation factor at specified height z_r	dB

5.2.10 Get Reflection Coefficient (GETREFCOEF) SU

The purpose of the GETREFCOEF SU is to compute the Fresnel complex reflection coefficient for a given grazing angle, ψ .

Upon entering, the proper dielectric constant nc_i^2 to be applied to the reflected ray must be determined. A DO WHILE loop is performed on array *rgrnd* to determine the index i at which the range $x_{reflect}$ (range at which ray is reflected) falls between two consecutive range points in *rgrnd*. Once this is found, the corresponding dielectric constant nc_i^2 is used in the following equations to compute the reflection coefficient:

$$\Gamma_V = \frac{nc_i^2 \sin(\psi) - \sqrt{nc_i^2 - \cos^2(\psi)}}{nc_i^2 \sin(\psi) + \sqrt{nc_i^2 - \cos^2(\psi)}},$$

$$\Gamma_H = \frac{\sin(\psi) - \sqrt{nc_i^2 - \cos^2(\psi)}}{\sin(\psi) + \sqrt{nc_i^2 - \cos^2(\psi)}}$$

where Γ_V and Γ_H represent the reflection coefficients for vertical and horizontal polarization, respectively, and nc_i^2 is computed in the DIEINIT SU and is given by

$$nc_i^2 = \varepsilon_i + j60\sigma_i\lambda.$$

ε_i and σ_i are the relative permittivity and conductivity, respectively, to be applied at range $rgrnd_i$, and λ is the wavelength.

If rough surface calculations are required ($ruf = \text{'true.'}$) the Miller-Brown roughness reduction factor is computed. Wind speeds specified by the calling CSCI are allowed to vary with range; therefore, an interpolation is performed to obtain the wind speed ω_s at the current range. The sea surface rms wave height is then computed from

$$ruf_{ht} = ruf_{fac} \omega_s^2,$$

and the roughness reduction factor ρ is determined by

$$\rho = \left(3.2x_g - 2 + \sqrt{(3.2x_g)^2 - 7x_g + 9} \right)^{-1/2}$$

where

$$x_g = \frac{1}{2} [ruf_{ht} \sin(\psi)]^2.$$

Table 80 and Table 81 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the GETREFCOEF SU.

Table 80. GETREFCOEF SU input data element requirements.

Name	Description	Units	Source
i_{flag}	Integer flag indicating what region reflection coefficient is being computed 0 = FE and RO regions 1 = PE region	N/A	Calling SU
i_{gr}	Number of different ground types specified	N/A	Calling CSCI
i_{pol}	Polarization flag: 0 = horizontal polarization 1 = vertical polarization	N/A	Calling CSCI
nc^2	Array of complex dielectric constants	N/A	DIEINIT SU
n_w	Number of wind speeds	N/A	Calling CSCI
ψ	Grazing angle	radians	Calling SU
$rgrnd$	Array containing ranges at which varying ground types apply.	meters	Calling CSCI
$rngwind$	Ranges of wind speeds entered: $rngwind_i$ = range of i^{th} wind speed	meters	Calling CSCI

Table 80. GETREFCOEF SU input data element requirements. (Continued)

Name	Description	Units	Source
<i>ruf</i>	Logical flag indicating if rough sea surface calculations are required 'true.' = perform rough sea surface calculations 'false.' = do not perform rough sea surface calculations	N/A	APMINIT CSC
<i>ruf_{fac}</i>	Factor used for wave height calculation	meters ⁻¹	APMINIT CSC
<i>ruf_{ht}</i>	Sea surface rms wave height	meters	APMINIT CSC
<i>wind</i>	Array of wind speeds	meters/sec	Calling CSCI
<i>X_{reflect}</i>	Range at which ray is reflected	meters	Calling SU

Table 81. GETREFCOEF SU output data element requirements.

Name	Description	Units
$\Gamma_{V,H}$	Complex reflection coefficient for vertical (V) and horizontal (H) polarization	N/A
R_{mag}	Magnitude of the reflection coefficient	N/A
φ	Phase of the reflection coefficient	radians

5.2.11 Mixed Fourier Transform (MIXEDFT) SU

The purpose of the MIXEDFT SU is to propagate the PE field in free space one PE range step, applying the Leontovich boundary condition, using the mixed Fourier transform as outlined by Kuttler and Dockery (Ref.8). For finite conducting boundaries (i.e., if vertical polarization is specified or rough surface calculations are required) and the frequency is less than 400 MHz, the central difference form of the DMFT is used. If the frequency is greater than 400 MHz, the backward difference form of the DMFT is used.

Upon entering the SU, the first and last elements of the w array w_0 and $w_{n_{ff}}$ are initialized to 0. If using the central difference form of the DMFT ($i_{alg} = 1$), the following steps 1 through 6 are performed.

1. The difference field for the vertical PE calculation grid is computed from the PE field as

$$w_i = \frac{U_{i+1} - U_{i-1}}{2\Delta z_{PE}} + \alpha_{h,v} U_i ; \quad i = 1, 2, 3, \dots, n_{m1}$$

2. Next, the FRSTP SU is referenced to transform w to p-space, propagate the field forward one PE range step, and is transformed back to z-space upon return.

3. The coefficients used in the central difference form of the mixed transform, c_{k1} and c_{k2} , are propagated to the new range as follows:

$$\begin{aligned} c_{k1} &= c_{k1} C_{1x} \\ c_{k2} &= c_{k2} C_{2x} \end{aligned}$$

4. The particular solution ym of Kuttler's difference equation is then computed as follows:

$$\begin{aligned} ym_i &= 2\Delta z_{PE} w_i + R_T w_{i-1}; \quad i = 1, 2, 3, \dots, n_{m1} \\ ym_0 &= 0, \end{aligned}$$

where R_T is a quadratic root as computed in the GETALN SU.

5. The complex PE field U is then determined from

$$\begin{aligned} U_{n-i} &= R_T (ym_{n-i} - U_{n-i+1}); \quad i = 1, 2, 3, \dots, n_{fft} \\ U_0 &= 0 \end{aligned}$$

6. The final step in computing the PE field U is

$$U_i = U_i + a_r r n_i + b_r r n_{n-i} (-1)^{n-i}; \quad i = 0, 1, 2, \dots, n_{fft},$$

where

$$\begin{aligned} a_r &= ck_1 - R_k \left(\frac{1}{2} U_0 + \frac{1}{2} U_{n_{fft}} r n_{n_{fft}} + \sum_{i=1}^{n_{fft}-1} U_i r n_i \right), \\ b_r &= ck_2 - R_k \left(\frac{1}{2} U_0 r n_{n_{fft}} + \frac{1}{2} U_{n_{fft}} + \sum_{i=1}^{n_{fft}-1} U_{n-i} r n_i (-1)^i \right). \end{aligned}$$

If using the backward difference form of the DMFT ($i_{alg} = 2$), then the following steps 1 through 6 are performed.

1. The difference field for the vertical PE calculation grid is computed from the PE field as

$$w_i = U_i - R_T U_{i-1}; \quad i = 1, 2, 3, \dots, n_{m1}$$

- Next, the FRSTP SU is referenced to transform w to p-space, propagate the field forward one PE range step, and is transformed back to z-space upon return.
- The coefficient $cmft$ is then propagated forward one range step via

$$cmft = cmft * cmft_x.$$

- The particular solution of the field U is now computed from

$$U_i = w_i + R_T U_{i-1}; \quad i = 1, 2, 3, \dots, n_{fft}$$

$$U_0 = 0.$$

- The coefficient a_r for the homogeneous solution is determined as

$$a_r = cmft - \sum_{i=0}^{n_{fft}-1} U_i r n_i.$$

- Finally, the PE field U for the backward difference DMFT is computed as the sum of the homogeneous and particular solutions:

$$U_i = U_i + a_r r n_i; \quad i = 0, 1, 2, \dots, n_{fft}.$$

Table 82 and Table 83 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the MIXEDFT SU.

Table 82. MIXEDFT SU input data element requirements.

Name	Description	Units	Source
$\alpha_{h,v}$	Surface impedance term for horizontal and vertical polarization	N/A	GETALN SU
C_{1x}	Constant used to propagate c_{k1} by one range step in central difference algorithm	N/A	GETALN SU
C_{2x}	Constant used to propagate c_{k2} by one range step in central difference algorithm	N/A	GETALN SU
ck_1	Coefficient used in central difference form of DMFT	N/A	ALN_INIT SU
ck_2	Coefficient used in central difference form of DMFT	N/A	ALN_INIT SU
$cmft$	Coefficient used in backward difference form of DMFT	N/A	ALN_INIT SU
$cmft_x$	Constant used to propagate $cmft$ by one range step in backward difference algorithm	N/A	GETALN SU

Table 82. MIXEDFT SU input data element requirements. (Continued)

Name	Description	Units	Source
i_{alg}	Integer flag indicating which DMFT algorithm is being used: 0 = no DMFT algorithm will be used 1 = use central difference algorithm 2 = use backward difference algorithm	N/A	APMINIT CSC
n_{fft}	Transform size	N/A	FFTPAR SU
n_{m1}	$n_{fft}-1$	N/A	APMINIT CSC
R_k	Coefficient used in C_1 and C_2 calculations.	N/A	GETALN SU
rn	Array of R_T to the i^{th} power (e.g., $rn_i = R_T^i$)	N/A	GETALN SU
R_T	Complex root of quadratic equation for mixed transform method based on Kuttler's formulation	N/A	GETALN SU
U	Complex field at range r	$\mu V/m$	PESTEP SU

Table 83. MIXEDFT SU output data element requirements.

Name	Description	Units
ck_1	Coefficient used in central difference form of DMFT	N/A
ck_2	Coefficient used in central difference form of DMFT	N/A
$cmft$	Coefficient used in backward difference form of DMFT	N/A
U	Complex field at range r	$\mu V/m$
w	Difference equation PE field array	$\mu V/m$
ym	Field from recursion equation for central difference DMFT	$\mu V/m$

5.2.12 Parabolic Equation Step (PESTEP) SU

The purpose of the PESTEP SU is to compute propagation loss at a specified range based upon the split-step Fourier PE algorithm.

Upon entering the PESTEP SU, if the current output range step, i_{stp} , is equal to 1, the current PE range r and r_{log} (10 times the logarithm of r) are set equal to zero. The current PE range step i_{PEstp} is also set equal to 0. An iterative DO WHILE loop is then begun to advance the PE solution such that for the current PE range, a PE solution is calculated from the solution at the previous PE range. This iterative procedure is repeated in the DO WHILE loop until r is greater than the output range r_{out} . The following steps (1 through 8) are performed for each PE range step within the DO WHILE loop.

1. First, if the current PE range, r , is greater than zero, then the height of the ground at the previous PE range y_{last} is set to the height of the ground at the current PE range y_{cur} . Next, the previous PE range r_{last} is set equal to the current PE range r . The

complex PE field, U , of the previous range is stored in array U_{last} for subsequent horizontal interpolation at range r_{out} . In addition, r is incremented by one PE range step, Δr_{PE} . A new r_{log} is computed and the PE range step i_{PEstp} is incremented by one. Finally, the range at which interpolation for range-dependent refractivity profiles is performed, r_{mid} , is also incremented by one-half the PE range step.

2. If performing a terrain case (f_{ter} is 'true.'), the ground heights, y_{cur} and y_{curm} , at range r and r_{mid} , respectively, must be determined. y_{cur} is set equal to the terrain height at the current PE range step, $tyh_{i_{PEstp}}$. y_{curm} is determined as

$$y_{curm} = \frac{1}{2} (tyh_{i_{PEstp}-1} + y_{cur}).$$

If y_{cur} is less than y_{last} , then the DOSHIFT SU is referenced to shift the field accordingly.

3. If the APM CSCI is being used in a range-dependent mode (i.e., the number of profiles n_{prof} is greater than 1), or if a terrain profile is specified, the REFINITER SU is referenced to compute a new modified refractive index profile, $profint$, adjusted by the local ground height y_{curm} at range r_{mid} . A new environmental phase array, $envpr$, is re-computed based on this new refractivity profile.
4. The current PE range is then checked against the range of the current ground type given by array $rgrnd$, and, if necessary, the ground type counter i_g is incremented. The GETALN SU is then referenced to compute a new surface impedance $\alpha_{h,v}$ if vertical polarization is required or if performing rough surface calculations.
5. In order to propagate the field in free space one PE range step, the MIXEDFT SU is referenced if the DMFT algorithm is required; otherwise, the FRSTP SU is referenced. The field is then multiplied by the environmental array $envpr$.
6. Next, if the current terrain slope is positive, the DOSHIFT SU is referenced to shift the field by the appropriate number of bins.
7. If XO calculations are to be performed ($i_{xo} \geq 1$) and the current PE range is greater than r_{alz} , then the FZLIM SU is referenced to determine and store the outward propagation angle at the top of the PE region for subsequent use in the EXTOSU.
8. Finally, after the output range r_{out} is reached and the DO WHILE loop exited, the CALCLOS SU is referenced to obtain the propagation loss values at the desired output heights at the current output range r_{out} .

Table 84 and Table 85 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the PESTEP SU.

Table 84. PESTEP SU input data element requirements.

Name	Description	Units	Source
Δr_{PE}	PE range step	meters	APMINIT CSC
Δr_{PE2}	1/2 PE range step	meters	APMINIT CSC
f_{ilt}	Cosine-tapered (Tukey) filter array	N/A	PEINIT SU
f_{ter}	Logical flag indicating if terrain profile has been specified: .true. = terrain profile specified .false. = terrain profile not specified	N/A	TERINIT SU
ψ	Array of interpolated grazing angles at each PE range step	radians	GRAZE_INT SU
i_{alg}	Integer flag indicating which DMFT algorithm is being used: 0 = no DMFT algorithm will be used 1 = use central difference algorithm 2 = use backward difference algorithm	N/A	APMINIT CSC
i_g	Counter indicating current ground type being modeled	N/A	APMINIT CSC
i_{gr}	Number of different ground types specified	N/A	Calling CSCi
i_{PE}	Number of PE range steps	N/A	PEINIT SU
i_{pol}	Polarization flag: 0 = horizontal polarization 1 = vertical polarization	N/A	Calling CSCi
i_{stp}	Current output range step index	N/A	Calling SU
i_{xo}	Number of range steps in XO calculation region	N/A	APMINIT CSC
iz	Counter for points stored in $ffacz$	N/A	FZLIM SU
iz_{inc}	Integer increment for storing points at top of PE region (i.e., points are stored at every iz_{inc} range step)	N/A	PEINIT SU
n_{fft}	PE Transform size	N/A	FFTPAR SU
n_{34}	$\frac{3}{4} n_{fft}$	N/A	PEINIT SU
n_4	$\frac{1}{4} n_{fft}$	N/A	PEINIT SU
n_{prof}	Number of refractivity profiles	N/A	Calling CSCi
PE_{flag}	Flag to indicate use of PE algorithm only: 'true.' = only use PE sub-model 'false.' = use automatic hybrid model	N/A	Calling CSCi
$profint$	Profile interpolated to every Δz_{PE} in height	M-units	REFINTER SU
r_{atz}	Range at which z_{lim} is reached (used for hybrid model)	meters	APMINIT CSC
$rgrnd$	Array containing ranges at which varying ground types apply.	meters	Calling CSCi
$rlog$	10 log(PE range r)	N/A	PESTEP SU

Table 84. PESTEP SU input data element requirements. (Continued)

Name	Description	Units	Source
r_{max}	Maximum specified range	meters	Calling CSCI
r_{out}	Current output range	meters	Calling SU
ruf	Logical flag indicating if rough sea surface calculations are required 'true.' = perform rough sea surface calculations 'false.' = do not perform rough sea surface calculations	N/A	APMINIT CSC
tyh	Adjusted height points of terrain profile at every PE range step.	meters	PEINIT SU
U	Complex PE field	$\mu\text{V/m}$	PESTEP SU
y_{cur}	Height of ground at current range r	meters	PESTEP SU
y_{last}	Height of ground at previous range r_{last}	meters	PESTEP SU

Table 85. PESTEP SU output data element requirements.

Name	Description	Units
$envpr$	Complex [refractivity] phase term array interpolated every Δz_{PE} in height	N/A
i_g	Counter indicating current ground type being modeled	N/A
j_{end}	Index at which valid propagation factor and loss values in $mpfl$ end	N/A
j_{start}	Index at which valid propagation factor and loss values in $mpfl$ begin	N/A
$mpfl$	2-dimensional propagation factor and loss array	cB
r_{last}	Previous PE range	meters
r_{log}	10 log (PE range r)	N/A
r_{loglst}	10 log (previous PE range r_{last})	N/A
r_{mid}	Range at which interpolation for range-dependent profiles is performed	meters
U	Complex PE field at range r	$\mu\text{V/m}$
U_{last}	Complex PE field at range r_{last}	$\mu\text{V/m}$
y_{cur}	Height of ground at current range r	meters
y_{curm}	Height of ground at range $r + \Delta r_{PE2}$	meters
y_{last}	Height of ground at previous range r_{last}	meters

5.2.13 Ray Trace (RAYTRACE) SU

Using standard ray trace techniques, a ray is traced from a starting height ant_{ref} and range 0, with a specified starting elevation angle, α , to a termination range, x_r . As the ray is being traced, an optical path length difference pl_d (the difference between the actual path length and x_r) and a derivative of range with respect to elevation angle, $dx/d\alpha$, are being continuously computed. If the ray should reflect from the surface, a grazing angle, ψ , is determined. Upon reaching the termination range, a terminal elevation angle, β , is determined along with a termination height, z_r .

The raytrace is conducted by stepping in profile levels and computing ending values. A number of stepping scenarios, based upon starting and ending elevation angles, determine the program flow of the RAYTRACE SU. These scenarios are a ray that is upgoing, a ray that is downgoing, and a ray that turns around within a layer.

Upon entering the SU, a running range, x_{sum} , the range at which a ray is reflected, $x_{reflect}$, $dxd\alpha$, pl_d , ψ , and a ray type (direct or reflected) flag, i_{type} , are initialized to zero. A temporary beginning elevation angle, a_{start} , is set equal to α , and an environmental profile level counter, i , is set equal to the array index for the height in the RO region corresponding to the transmitter height, i_{start} .

The sub-steps within the following steps (1 and 2) are now repeated while x_{sum} remains less than the termination range x_r . Upon failure to meet this repetition criterion, the SU program flow continues with step 3 below.

1. The beginning angle a_{start} is examined to determine if the ray is initially upgoing (i.e., $a_{start} \geq 0$) or downgoing. If it is upgoing, the SU program flow continues with steps 1.a through 1.e; otherwise the program flow continues with step 2 below.
- a. The level counter is examined and if the ray is in the highest layer, (i.e., $i = l_{levels}$), the ending angle, height, range/angle derivative and path length difference are given as

$$\beta = a_{start} + (x_r - x_{sum})gr_i,$$

$$z_r = zrt_i + \frac{\beta^2 - a_{start}^2}{2 gr_i},$$

$$dxd\alpha = dxd\alpha + \frac{1}{gr_i} \left(\frac{\alpha}{\beta} - \frac{\alpha}{a_{start}} \right),$$

$$pl_d = pl_d + \frac{1}{gr_i} \left[\left(rm_i - \frac{a_{start}^2}{2} \right) (\beta - a_{start}) + \frac{1}{3} (\beta^3 - a_{start}^3) \right],$$

respectively, where gr is an intermediate M-unit gradient, rm is an intermediate M-unit, and zrt is an intermediate height. Satisfaction of this condition causes the failure of the repetition criterion, and the SU program flow continues with step 3 below.

- b. If the ray is not in the highest layer, the ray must be examined to determine if it will turn around and become a downgoing ray within the current layer. This is done by looking at the radical term, rad , which will be used in the ending angle calculation. The radical term is given as

$$rad = a_{start}^2 + q_i$$

where q is an intermediate M-unit difference. If rad is greater than or equal to zero, a solution for the ending angle is possible. The ray will not turn around and the program flow continues with step 1.c, otherwise the program flow continues with step 1.d.

- c. Before calculations can continue, the possible ending range must be compared to the termination range. This possible ending range is determined as

$$x_{temp} = x_{sum} + \frac{\beta - a_{start}}{gr_i},$$

$$\beta = \sqrt{rad}.$$

This possible ending range is compared to the termination range, and, if it is larger, the ending angle, height, range/angle derivative, and path length difference are computed from equations given in step 1.a. Satisfaction of this condition causes the failure of the repetition criterion, and the SU program flow continues with step 3 below.

If the ray has not reached the termination range, x_{sum} is updated to x_{temp} ; the range/angle derivative and path length difference are computed from equations given in step 1.a, where $\beta = \sqrt{rad}$; a_{start} is updated to β ; the level counter is incremented by one; and the program flow returns to step 1 above.

- d. If the ray has, in fact, turned around within the current layer, a determination must be made for the ray reaching a full range step within the still upgoing segment, for the ray reaching a full range step within the downgoing segment, or the ray exceeding the termination range. The full range step is given by

$$x_{temp} = x_{sum} - \frac{a_{start}}{gr_i},$$

which is compared to the termination range. If it exceeds the termination range, the ending angle, the ending height, the range/angle derivative, and the path

length difference are determined from equations given in step 1.a. Satisfaction of this condition causes the failure of the repetition criterion, and the SU program flow continues with step 3 below. If the termination range has not been exceeded, further examination of the ray's segments must be made.

- e. At this point, x_{sum} is updated to x_{temp} ; x_{temp} is recalculated as shown in step 1.d; and x_{temp} is again compared to the termination range. If the termination range has been exceeded, the ending angle is given as

$$\beta = (x_r - x_{sum})gr_i,$$

and the ending height, the range/angle derivative, and the path length difference are now determined from equations shown in step 1.a. Satisfaction of this condition causes the failure of the repetition criterion, and the SU program flow continues with step 3 below.

If the termination range has not been exceeded, x_{sum} is updated to x_{temp} ; β is updated to $-a_{start}$; the range/angle derivative and path length difference are determined from equations shown in step 1.a; a_{start} is updated to β ; and the program flow returns to step 1 above.

2. **Note!** The equations for the upgoing ray within step 1 above apply equally to the downgoing ray except where specified otherwise. However, in applying these equations to step 2, the level counter, i , within the intermediate M-unit gradient sub-term, gr , must be reduced by one.

The beginning angle a_{start} has been examined in step 1 above and the ray has been determined to be initially downgoing. Similar to step 1 above, the ray must be examined to determine if it has turned around and has become an upgoing ray within the current layer. This is done by looking at the radical term, rad , which will be used in the ending angle calculation. This radical term is given as

$$rad = a_{start}^2 - q_{i-1}.$$

If rad is greater than or equal to zero, a solution for the ending angle is possible. The ray has not turned around and the program flow continues with steps 2.a through 2.c below, otherwise the program flow continues with step 2.d.

- a. Before calculations can continue, the possible ending range must be compared to the termination range. This possible ending range is determined from the equation given for x_{temp} in step 1.c, where β is now $-\sqrt{rad}$. This possible ending range is compared to the termination range and if it is larger, the ending angle, the

ending height, the range/angle derivative, and the path length difference are computed from equations shown in step 1.a. Satisfaction of this condition causes the failure of the repetition criterion, and the SU program flow continues with step 3 below.

- b. If the termination range has not been exceeded, x_{sum} is updated to x_{temp} ; the range/angle derivative and path length difference are computed as shown in step 1.a, where $\beta = -\sqrt{rad}$; a_{start} is updated to β ; and the level counter is decremented by one.
- c. The level counter is examined and if it is zero, the ray has reflected from the surface. In this case, the ray type flag is set to 1 to indicate a reflection, the grazing angle is set as $\psi = |a_{start}|$, and $x_{reflect}$ is set equal to x_{temp} . At this point, a symmetry check is made. The idea of symmetry says that the ray will return to its starting height, at twice the reflection range, with an ending elevation angle opposite the starting elevation angle. Symmetry is used for APM speed efficiency so as to preclude redundant ray trace calculations on the upward path back to the starting height. Prior to applying symmetry, however, the possible ending range (twice x_{sum}) must be compared to the termination range. If the termination range is exceeded by making the symmetry assumption, a_{start} is updated to $-a_{start}$ and the assumption is vacated. If not, however, the assumption is invoked and a_{start} is updated to $-\alpha$; x_{sum} , $dx d\alpha$, and pl_d , are doubled; and the level counter is restored to i_{start} . Control is now returned to the top of step 1 above.
- d. From step 2, the ray has turned around within the current layer and is now an upgoing ray. Similar to the upgoing case of step 1, a determination must be made for the ray reaching a full range step within the still downgoing segment, for the ray reaching a full range step within the upgoing segment, or the ray exceeding the termination range. The full range step is given by x_{temp} as computed step 1.d.

If the full range step exceeds the termination range, the ending angle, the ending height, the range/angle derivative, and the path length difference are computed from equations shown in step 1.a. Satisfaction of this condition causes the failure of the repetition criterion, and the SU program flow continues with section 3 below. If the termination range has not been exceeded, further examination of the ray's segments must be made.

- e. At this point, x_{sum} is updated to x_{temp} ; x_{temp} is recalculated as in step 1.d, and x_{temp} is again compared to the termination range. If the termination range has been exceeded, the ending angle is determined as in step 1.e; the ending height, range/angle derivative, and path length difference are determined as in step 1.a. Satisfaction of this condition causes the failure of the repetition criterion, and the SU program flow continues with step 3 below.

If the termination range has not been exceeded, x_{sum} is updated to x_{temp} ; β is updated to $-a_{start}$; the range/angle derivative and path length difference are determined as in step 1.a; a_{start} is updated to β ; and the program flow returns to step 1 above.

3. Within APM, the terminal elevation angle is not allowed to be equal to zero. Therefore, if its absolute value is less than 10^{-10} , it is reset to 10^{-10} while retaining its present sign.

Table 86 and Table 87 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the RAYTRACE SU.

Table 86. RAYTRACE SU input data element requirements.

Name	Description	Units	Source
α	Source elevation angle	radians	Calling SU
gr	Intermediate M-unit gradient array, RO region	(M-unit/m) 10^{-6}	REFINIT SU
i_{start}	Array index for height in RO region corresponding to ant_{ref}	N/A	REFINIT SU
$levels$	Number of levels in gr , q and zrt arrays	N/A	REFINIT SU
q	Intermediate M-unit difference array, RO region	$2(M\text{-unit})10^{-6}$	REFINIT SU
rm	Intermediate M-unit array, RO region	M-unit 10^{-6}	REFINIT SU
x_r	Terminal range - called x_{ROn} in ROCALC SU, equivalent to r_{out} in calling SU	meters	Calling SU
zrt	Intermediate height array, RO region	meters	REFINIT SU

Table 87. RAYTRACE SU output data element requirements..

Name	Description	Units
β	Terminal elevation angle	radians
$dxd\alpha$	Derivative of range with respect to elevation angle	meters/radians
i_{type}	Ray type (direct or reflected) flag	N/A
pl_d	Path length from range x_r	meters
ψ	Grazing angle	radians
$x_{reflect}$	Range at which ray is reflected	meters
z_r	Terminal height	meters

5.2.14 Refractivity Interpolation (REFINTER) SU

The purpose of the REFINTER SU is to interpolate both horizontally and vertically on the modified refractivity profiles. Profiles are then adjusted so they are relative to the local ground height .

Upon entry, the number of height/refractivity levels , $lvlep$, for the current profile is set equal to the user-specified number of levels for all profiles specified, $lvlp$. For the range-dependent case, all profiles have the same number of levels.

If there is a range-dependent environment (i.e., $n_{prof} > 1$), horizontal interpolation to range r_{ange} is performed between the two neighboring profiles that are specified relative to mean sea level. In this case, the following calculations are made. If r_{ange} is greater than the range for the next refractivity profile rv_2 , then the index j (indicating the range of the previous refractivity profile) is set equal to the counter for the range of the current profile i_s ; i_s is then incremented by one. Next, the range of the previous refractivity profile rv_1 is set equal to rv_2 , and rv_2 is set equal to the range of the i_s^{th} profile, $rngprof_{i_s}$. The fractional range fv for the interpolation is given by

$$fv = \frac{r_{ange} - rv_1}{rv_2 - rv_1} .$$

The array $refdum$, containing M-unit values for the current (interpolated) profile and the array $htdum$ containing height values for the current (interpolated) profile are determined from

$$\begin{aligned} refdum_i &= refmsl_{i,j} + fv(refmsl_{i,i_s} - refmsl_{i,j}); \quad i = 1,2,3,...lvlep \\ htdum_i &= hmsl_{i,j} + fv(hmsl_{i,i_s} - hmsl_{i,j}); \quad i = 1,2,3,...lvlep \end{aligned}$$

where $refmsl$ and $hmsl$ are 2-dimensional arrays containing refractivity and height, respectively, with respect to mean sea level of each user-specified profile.

The REMDUP SU is referenced to remove duplicate refractivity levels, with $lvlep$ being the number of points in the profile at range r_{ange} . The PROFREF SU is then referenced to adjust the new profile (i.e., $refdum$ and $htdum$) relative to the internal reference height h_{minter} , corresponding to the minimum height of the terrain profile. The PROFREF SU is then referenced once more to adjust the profile relative to the local ground height y_{curm} , and upon exit from the PROFREF SU, the INTPROF SU is referenced to interpolate vertically on the refractivity profile at each PE mesh height point. This results in the n_{fft} -point profile array $profint$ containing the interpolated M-unit values for the refractivity at r_{ange} , where n_{fft} is the transform size.

Upon exiting the REFINTER SU, rv_1 and index j are saved for use upon the next reference of the SU.

Table 88 and Table 89 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the REFINTER SU.

Table 88. REFINTER SU input data element requirements.

Name	Description	Units	Source
f_{ter}	Logical flag indicating if terrain profile has been specified: .true. = terrain profile specified .false. = terrain profile not specified	N/A	TERINIT SU
h_{minter}	Minimum height of terrain profile	meters	TERINIT SU
$hmsl$	2-dimensional array containing heights with respect to mean sea level of each profile. Array format must be $hmsl_{i,j}$ = height of i^{th} level of j^{th} profile. $j=1$ for range-independent cases	meters	Calling CSCI
i_s	Counter for current profile	N/A	REFINIT SU REFINTER SU
i_{stp}	Current output range step index	N/A	Calling SU
$lvlp$	Number of height/refractivity levels in profiles	N/A	Calling CSCI
n_{prof}	Number of refractivity profiles	N/A	Calling CSCI
r_{ange}	Range for profile interpolation	meters	Calling SU
$refmsl$	2-dimensional array containing refractivity with respect to mean sea level of each profile. Array format must be $refmsl_{i,j}$ = M-unit at i^{th} level of j^{th} profile. $j=1$ for range-independent cases	M-unit	Calling CSCI
$rngprof$	Ranges of each profile. $rngprof_i$ = range of i^{th} profile	meters	Calling CSCI
rv_2	Range of the next refractivity profile	meters	REFINIT SU REFINTER SU
y_{cum}	Height of ground midway between last and current PE range	meters	PESTEP SU

Table 89. REFINTER SU output data element requirements.

Name	Description	Units
$htdum$	Height array for current interpolated profile	meters
i_s	Counter for current profile	N/A
$lvlep$	Number of height/refractivity levels in profile $refdum$ and $htdum$	N/A
$profint$	Profile interpolated to every Δz_{PE} in height	M-units
$refdum$	M-unit array for current interpolated profile	M-units
rv_2	Range of the next refractivity profile	meters

5.2.15 Ray Optics Calculation (ROCALC) SU

The purpose of the ROCALC SU is to compute the RO components which will be needed (by the ROLOSS SU) in the calculation of propagation loss at a specified range and height within the RO region. These components are the magnitudes for a direct-path and surface-reflected ray, Fd^2 and Fr^2 , respectively, and the total phase lag angle, Ω , between the direct-path and surface-reflected rays.

The RO region may be visualized as having a grid of points superimposed upon it. The grid points are defined at the intersection of a series of lines sloping upward from the origin and a series of vertical lines at varying ranges. The grid point counter k and the vertical lines are defined at varying ranges, two of which are represented by the terms x_{ROp} , a range for which the RO calculations were previously performed, and x_{ROn} , the next calculation range.

The sloping line with the greatest angle (indicated by $k = k_{max}$) is a function of the maximum APM output height, ht_{ydif} , adjusted for terrain and reference heights, and the next calculation range. The sloping line with the least angle (indicated by $k = k_{minp}$) is a function of the height at the top of the PE region and the range of the previous RO calculations.

The following steps 1 through 4 are performed while the current range, x , is greater than x_{ROn} .

1. The terms of Table 90 (defined in Table 91) are initialized or updated based upon the RO calculation range counter i_{ROp} . If i_{ROp} equals -1, the terms are initialized; otherwise, the terms are updated. It should be noted that the terms must be computed in the order they appear in the table to ensure proper values are assigned to component terms.

Table 90. RO region indices, angles, and ranges.

For $i_{ROp} = -1$ (initialize terms)	For $i_{ROp} \neq -1$ (update terms)
$i_{ROp} = 1$	$i_{ROp} = 1 - i_{ROp}$
$i_{ROn} = 0$	$i_{ROn} = 1 - i_{ROn}$
N/A	$x_{ROp} = x_{ROn}$
$k_{minp} = 0$	$k_{minp} = k_{minn}$
$k_{minn} = 0$	$k_{minn} = 0$
$ht_{ydif} = ht_{lim} - y_{fref}$	N/A
$d\alpha = \text{MIN}\left(\frac{\mu_{bwr}}{2}, .01745\right)$	N/A

Table 90. RO region indices, angles, and ranges. (Continued)

For $iROp = -1$ (initialize terms)	For $iROp \neq -1$ (update terms)
$k_{max} = 88$	$k_{max} = \text{MIN} \left(88, \text{INT} \left(\frac{1000 h t_{ydif}}{x_{ROp}} \right) + 2 \right)$
$frac_{RO} = 0$	$frac_{RO} = \left(\text{MAX} \left(\frac{.001 k_{max}}{d\alpha}, 5 \right) - 1 \right)^{-1}$; for $frac_{RO} < .25$
N/A	$\Delta x_{RO} = frac_{RO} x_{ROp}$
$x_{ROn} = x$	$x_{ROn} = x_{ROp} + \Delta x_{RO}$

2. To calculate the RO components at each vertical point for the next range, x_{ROn} , a ray trace within a Newton iteration method is used to find a direct-path ray and a surface-reflected ray which will both originate at the transmitter height, ant_{ref} , and terminate at the same grid point, z_k . The results of the iteration are examined and if either of the rays has not been found, an adjustment in the lower boundary of the RO region is made. Following the conclusion of the iterations, the antenna pattern factors for each ray are obtained, a surface reflection coefficient for the surface-reflected ray is computed, and the RO components are calculated.

Prior to all calculations for each vertical point, the ray trace must be initialized with beginning direct-path and surface-reflected ray elevation angles, α_d and α_r , respectively; and derivatives of height with respect to these elevation angles, $dzd\alpha_d$ and $dzd\alpha_r$. A starting assumption is made that the direct-path ray and the surface-reflected rays are parallel to each other. Thus, α_d is initialized as $0.001 k_{max}$ and α_r is initialized as $-\alpha_d$. The RAYTRACE SU is referenced separately with α_d and α_r to obtain termination elevation angles, β_d and β_r , and the two derivatives of range with respect to elevation angle, $dx d\alpha_d$ and $dx d\alpha_r$, which are used in turn to compute the needed derivatives of height with respect to elevation angle given as $-\beta_d dx d\alpha_d$ and $-\beta_r dx d\alpha_r$.

3. Once the raytrace has been initialized, the following steps 3.a through 3.f are performed for each vertical grid point, z_k , beginning with $k = k_{max}$ and subsequently decrementing k downward while k remains $\geq k_{minn}$. Once k has reached zero, processing continues with step 4 below.

- a. The termination height is computed as

$$z_k = x_{ROn} \cdot 0.001 \cdot k$$

where k is the grid point counter.

- b. The Newton iteration method to find the direct path ray from ant_{ref} to z_k is started. This iteration is continued until the difference between the ray trace ending height z_d and z_k is less than a height difference tolerance z_{tol} ; but in any case, no more than 10 times. The direct-path elevation angle is given as

$$\alpha_d = \alpha_d - \frac{z_d - z_k}{dzd\alpha_d}$$

where z_d and $dzd\alpha_d$ are obtained from the ray trace initialization of step 2 above for the first iteration and from the previous iteration for subsequent iterations.

The RAYTRACE SU is referenced and a new $dzd\alpha_d$ is calculated as $-\beta_d dx d\alpha_d$. This new $dzd\alpha_d$ is examined and if it is less than 10^{-6} , or if the ray type flag i_{type} , returned from the RAYTRACE SU, indicates the ray has reflected, the lower boundary of the RO region is adjusted by setting k_{minn} equal to one more than k , and the iteration for the direct ray is stopped.

- c. The Newton iteration method to find the surface-reflected ray from ant_{ref} to z_k is now started. This iteration should be continued until the difference between the ray trace ending height, z_r and z_k is less than a height difference tolerance z_{tol} ; but in any case, no more than 10 times. The reflected-path elevation angle is given as

$$\alpha_r = \alpha_r - \frac{z_r - z_k}{dzd\alpha_r}$$

where z_r and $dzd\alpha_r$ are obtained from the ray trace initialization of step 2 above for the first iteration and from the previous iteration for subsequent iterations.

The RAYTRACE SU is referenced and a new $dzd\alpha_r$ is calculated as $-\beta_r dx d\alpha_r$. This new $dzd\alpha_r$ is examined and if it is less than 10^{-6} , or if i_{type} indicates the ray is a direct ray, the lower boundary of the RO region is adjusted by setting k_{minn} equal to one more than k , and the iteration for the surface-reflected ray is stopped.

- d. A test is made to determine if the grazing angle, ψ , (returned from the RAYTRACE SU) is less than the limiting value, ψ_{lim} , and if so, the lower boundary of the RO region is adjusted by setting k_{minn} equal to k .

- e. The magnitudes for the direct-path and surface-reflected ray, Fd^2 and Fr^2 respectively, are now given as

$$Fd^2 = \left| \frac{x_{ROn}}{dzd\alpha_d} \right| f^2(\alpha_d),$$

$$Fr^2 = \left| \frac{x_{ROn}}{dzd\alpha_r} \right| [f(\alpha_r)R_{mag}]^2,$$

where the amplitude of the surface reflection coefficient, R_{mag} , is obtained from a reference to the GETREFCOEF SU; the antenna pattern factors $f(\alpha_d)$ and $f(\alpha_r)$ are obtained from references to the ANTPAT SU; and the derivatives of height with respect to elevation angle are obtained from the RAYTRACE SU within the Newton iteration of steps 3.b and 3.c above.

- f. The total phase lag between the direct-path and surface-reflected rays is computed as

$$\Omega = (pl_r - pl_d)k_o + \varphi,$$

where the ray path lengths pl_d and pl_r are obtained from the RAYTRACE SU within the Newton iteration of steps 3.b and 3.c above; the reflection coefficient phase lag angle, φ , is obtained from a reference to the GETREFCOEF SU; and k_o is the free-space wave number.

4. If the point counter k has been reduced to zero by the procedures of steps 3.a through 3.f above, the surface values of magnitudes for the direct-path and surface-reflected rays are both set equal to the last value of Fd^2 and the total phase lag between the direct-path and surface-reflected rays is set equal to π . If vertical polarization has been specified, then a further calculation is performed for the special case when the RO output height is less than 0 due to the height adjustment of y_{fref} . A FE calculation is made at this lowest height and replaces the previous values of magnitudes for the direct and surface-reflected paths, along with a new phase lag.

Table 91 and Table 92 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the ROCALC SU.

Table 91. ROCALC SU input data element requirements.

Name	Description	Units	Source
ant_{ref}	Transmitting antenna height relative to the reference height h_{minter}	meters	TERINIT SU
μ_{bwr}	Antenna vertical beamwidth	radians	Calling CSCI
$d\alpha$	$\frac{1}{2} \mu_{bwr}$	radians	ROCALC SU
$frac_{RO}$	RO range interval fraction (0.0 to 0.25)	N/A	ROCALC SU
ht_{lim}	Maximum height relative to h_{minter}	meters	TERINIT SU
ht_{ydif}	$ht_{lim} - y_{fref}$	meters	ROCALC SU
i_{pol}	Polarization flag: 0 = horizontal polarization 1 = vertical polarization	N/A	Calling CSCI
i_{ROn}	Array index for next range in RO region	N/A	ROCALC SU
i_{ROp}	Array index for previous range in RO region	N/A	APMINIT CSC ROCALC SU
k_o	Free-space wave number	meters ⁻¹	APMINIT CSC
k_{minn}	Array index for minimum angle in RO region at range x_{ROn}	N/A	ROCALC SU
ψ_{lim}	Grazing angle of limiting ray	radians	APMINIT CSC
$twoka$	Twice the effective earth radius	meters	GET_K SU
x	Current range	meters	Calling SU
x_{ROn}	Next range in RO region	meters	APMINIT CSC
x_{ROp}	Previous range in RO region	meters	ROCALC SU
y_{fref}	Ground elevation height at source	meters	APMINIT CSC
$zoutma$	Array output heights relative to "real" ant_{ref}	meters	APMINIT CSC
$zoutpa$	Array output heights relative to "image" ant_{ref}	meters	APMINIT CSC
zro	Array of output heights	meters	APMINIT CSC
z_{tol}	Height tolerance for Newton's method	meters	APMINIT CSC

Table 92. ROCALC SU output data element requirements.

Name	Description	Units
$d\alpha$	$\frac{1}{2} \mu_{bwr}$	radians
Δx_{RO}	RO range interval	meters
Fd^2	Magnitude array, direct ray	N/A
Fr^2	Magnitude array, reflected ray	N/A
$frac_{RO}$	RO range interval fraction (0.0 to 0.25)	N/A
ht_{ydif}	$ht_{lim} - y_{fref}$	meters
i_{ROn}	Array index for next range in RO region	N/A
i_{ROp}	Array index for previous range in RO region	N/A
k_{max}	Array index for maximum angle in RO region at range x_{ROn}	N/A

Table 92. ROCALC SU output data element requirements. (Continued)

Name	Description	Units
k_{minn}	Array index for minimum angle in RO region at range x_{ROn}	N/A
k_{minp}	Array index for minimum angle in RO region at range x_{ROp}	N/A
Ω	Total phase angle array	radian
x_{ROn}	Next range in RO region	meters
x_{ROp}	Previous range in RO region	meters

5.2.16 Ray Optics Loss (ROLOSS) SU

The purpose of the ROLOSS SU is to calculate propagation factor and loss values at all valid RO heights at a specified range based upon the components of magnitude for a direct-path and surface-reflected ray, Fd^2 and Fr^2 respectively, and the total phase lag angle, Ω , between the two rays as determined by the ROCALC SU.

Upon entering the SU, the ROCALC SU is referenced to obtain the current values of the direct and reflected ray magnitudes, along with the phase lag for all heights at the specified range r_{out} .

For purposes of computational efficiency, an interpolation from the magnitude and total phase lag angle arrays established by the ROCALC SU is made to obtain these three quantities at each APM vertical output mesh point within the RO region.

From the interpolated phase lag angle and ray magnitudes, a propagation factor is calculated which is used in turn with the free-space propagation loss, to obtain a propagation loss at each vertical APM output point.

A range ratio term to be used within the interpolation scheme is defined as

$$ratio = \frac{r_{out} - x_{ROp}}{\Delta x_{RO}}.$$

The phase lag angle and ray magnitude arrays have been filled at grid points defined by a series of sloping lines and the next and previous RO calculation range, x_{ROn} and x_{ROp} , respectively. Which values to interpolate from are determined by the sloping line immediately above and the sloping line immediately below the current APM output point of interest. To begin the calculations, k_{lo} is initialized to k_{max} , the line with the greatest angle.

The following steps 1 through 4 are now taken, decrementing downward in APM output points from the maximum output height index in the RO region, j_{max} , to the

minimum output height index, in the RO region, j_{min} where the index j varies from j_{max} to j_{min} .

1. Interpolation of Fd^2 , Fr^2 , and Ω values occurs in two stages. The first stage is horizontally, above and below the APM output point (i.e., along the lines k_{lo} and k_{hi}). These values will be used in turn, in a vertical interpolation stage to obtain values at the APM output point itself. It may be, however, that more than one APM output point will fall between two adjacent k lines. In this case, it would be redundant to perform the horizontal interpolation more than once. For this reason, a temporary k line counter is established which will be used in comparison with k_{lo} to determine if interpolation is necessary or if the previously interpolated horizontal values may again be used in the vertical interpolation. This temporary k counter is given by

$$k_{temp} = \text{INT} \left(\frac{1000 zro_j}{r_{out}} \right)$$

where j is the APM output point counter, and zro_j is the j^{th} output height point. If k_{temp} is less than the current k_{lo} , the APM output point occurs below the current lower k line and horizontal interpolations must be performed using the following steps 1.a through 1.c; otherwise, the horizontal interpolations are unnecessary and the SU may proceed with step 2.

- a. The lower k line, k_{lo} , is reset to k_{temp} and the upper k line, k_{hi} , is set to one more than k_{lo} .
- b. In preparation for the interpolation, component terms (horizontal differences of direct and surface-reflected magnitudes and phase lag angles) along the k_{lo} and k_{hi} lines are given as

$$\begin{aligned} \Delta Fd_{lo}^2 &= Fd_{i_{ROn},k_{lo}}^2 - Fd_{i_{ROp},k_{lo}}^2, \\ \Delta Fr_{lo}^2 &= Fr_{i_{ROn},k_{lo}}^2 - Fr_{i_{ROp},k_{lo}}^2, \\ \Delta \Omega_{lo} &= \Omega_{i_{ROn},k_{lo}} - \Omega_{i_{ROp},k_{lo}}, \end{aligned}$$

substituting the index k_{hi} for k_{lo} as appropriate. It should be noted that these horizontal differences need only be calculated while both k_{hi} and k_{lo} remain greater than or equal to both k_{minp} and k_{minn} . If these conditions are not met, any continued difference calculations would take place within the PE region which would yield undesirable results. For failure of these conditions, the previously calculated difference values are used for the lower RO region boundary calculations.

- c. If k_{lo} is greater than or equal to k_{minp} , the horizontally interpolated direct and surface-reflected magnitudes and phase lag angles along the k_{lo} line can proceed in a forward manor (from x_{ROp} to r_{out}). These values are given as

$$\begin{aligned} Fd_{lo}^2 &= Fd_{i_{ROp},k_{lo}}^2 + ratio \Delta Fd_{lo}^2, \\ Fr_{lo}^2 &= Fr_{i_{ROp},k_{lo}}^2 + ratio \Delta Fr_{lo}^2, \\ \Omega_{lo} &= \Omega_{i_{ROp},k_{lo}} + ratio \Delta \Omega_{lo}. \end{aligned}$$

In a like manor, the same three equations above are used to get the values along the k_{hi} line by substituting the index k_{hi} , assuming, however, k_{hi} is also greater than or equal to k_{minp} . Should either k_{lo} or k_{hi} be less than k_{minp} , the interpolation must proceed in a backward manor (from x_{ROn} to r_{out}). The above three equations may again be used by substituting the index i_{ROn} for i_{ROp} and the value $(1 - ratio)$ for $ratio$.

2. Once the horizontal interpolation of magnitudes and phase lag angles has been accomplished, the vertical interpolation of magnitudes and phase lag angles at the APM output point may proceed as

$$\begin{aligned} Fd^2 &= Fd_{lo}^2 + ratio_k (Fd_{hi}^2 - Fd_{lo}^2), \\ Fr^2 &= Fr_{lo}^2 + ratio_k (Fr_{hi}^2 - Fr_{lo}^2), \\ \Omega &= \Omega_{lo} + ratio_k (\Omega_{hi} - \Omega_{lo}), \end{aligned}$$

Where ratio from k_{lo} to k_{hi} is

$$ratio_k = \frac{1000 zro_j}{r_{out}} - k_{lo}.$$

3. From the magnitudes of the direct and surface-reflected components and the phase lag angle, the square of the propagation factor at the APM output point is given as

$$F^2 = \left| Fd^2 + Fr^2 + 2\sqrt{|Fd^2 Fr^2|} \cos \Omega \right|,$$

which, in turn, is converted to a propagation factor expressed in decibels by

$$F_{dB} = 10 \text{LOG}_{10} [\text{MAX}(F^2, 10^{-25})].$$

4. Finally, the total propagation loss and propagation factor is computed at the j^{th} APM output point and converted to centibels accordingly:

$$L_{dB} = fsl_{i_{stp}} - F_{dB} + gas_{loss} ,$$

$$F_{dB} = fsl_{i_{stp}} - L_{dB} ,$$

$$mpfl_{1,j} = \text{NINT}(10 L_{dB}) ,$$

$$mpfl_{2,j} = \text{NINT}(10 F_{dB}) ,$$

where $fsl_{i_{stp}}$ is the free-space loss at the i_{stp}^{th} output range.

Table 93 and Table 94 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the ROLOSS SU. Table 95 identifies terms which are used internal to the ROLOSS SU and whose value must be retained from SU call to SU call for reasons of computational efficiency.

Table 93. ROLOSS SU input data element requirements.

Name	Description	Units	Source
Δx_{RO}	RO range interval	meters	ROCALC SU
Fd^2	Magnitude array, direct ray	N/A	ROCALC SU
Fr^2	Magnitude array, reflected ray	N/A	ROCALC SU
fsl	Free space loss array for output ranges	dB	APMINIT CSC
gas_{loss}	Gaseous absorption loss at range r_{out}	dB	APMSTEP CSC
i_{ROn}	Array index for next range in RO region	N/A	ROCALC SU
i_{ROp}	Array index for previous range in RO region	N/A	ROCALC SU
i_{stp}	Current output range step index	N/A	Calling SU
j_{max}	Array index for maximum output height in RO region	N/A	Calling SU
j_{min}	Array index for minimum output height in RO region	N/A	Calling SU
k_{max}	Array index for maximum angle in RO region at range x_{ROn}	N/A	ROCALC SU
k_{minn}	Array index for minimum angle in RO region at range x_{ROn}	N/A	ROCALC SU
k_{minp}	Array index for minimum angle in RO region at range x_{ROp}	N/A	ROCALC SU
Ω	Total phase angle array	radians	ROCALC SU
r_{out}	Current output range	meters	Calling SU
x_{ROp}	Previous range in RO region	meters	ROCALC SU
zro	Array of output heights in RO region	meters	APMINIT CSC

Table 94. ROLOSS SU output data element requirements.

Name	Description	Units
<i>mpfl</i>	Propagation factor and loss array	cB

Table 95. ROLOSS SU save data element requirements.

Name	Description	Units	Source
$\Delta\Omega_{hi}$	Difference in total phase lag angle along Δx_{RO} above desired APM output point	radians	ROLOSS SU
$\Delta\Omega_{lo}$	Difference in total phase lag angle along Δx_{RO} below desired APM output point	radians	ROLOSS SU
ΔFd_{lo}^2	Difference in direct ray magnitude along Δx_{RO} below desired APM output point	N/A	ROLOSS SU
ΔFd_{hi}^2	Difference in direct ray magnitude along Δx_{RO} above desired APM output point	N/A	ROLOSS SU
ΔFr_{lo}^2	Difference in reflected ray magnitude along Δx_{RO} below desired APM output point	N/A	ROLOSS SU
ΔFr_{hi}^2	Difference in reflected ray magnitude along Δx_{RO} above desired APM output point	N/A	ROLOSS SU

5.2.17 Save Profile (SAVEPRO) SU

The purpose of the SAVEPRO SU is to store the gradients and heights of the current refractivity profile, upon each reference to the FZLIM SU, from the top of the PE calculation region to the maximum user-specified height.

Upon entering, the current profile height array *htdum* is searched to find the index *i* such that *htdum_i* is the first height in the profile that is greater than the maximum PE calculation height, *z_{lim}*. The counter *l_{new}* is then initialized to -1.

Next, the gradients are calculated and stored, along with corresponding heights, as follows

$$grad_{l_{new},iz} = \frac{refdum_{j+1} - refdum_j}{htdum_{j+1} - htdum_j},$$

$$htr_{l_{new},iz} = htdum_j,$$

where *j* is incremented by one from *i* to *lvlep-1*, *l_{new}* is incremented by one with each increment in *j*, and *iz* represents the range step index for XO calculations.

Before exiting, the last height level in *htdum* is stored and the final number of levels, l_{new} , in the iz^{th} profile (represented by *grad* and *htr*) is stored in array *lvl*.

Table 96 and Table 97 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the SAVEPRO SU.

Table 96. SAVEPRO SU input data element requirements.

Name	Description	Units	Source
<i>htdum</i>	Height array for current profile	meters	REFINTER SU
<i>iz</i>	Number of calculation range steps for XO region	N/A	FZLIM SU
<i>lvl</i>	Number of height/refractivity levels in profile <i>refdum</i> and <i>htdum</i>	N/A	REFINTER SU
<i>refdum</i>	M-unit array for current profile	M-units	REFINTER SU
z_{lim}	Maximum height in PE calculation region	meters	FFTPAR SU

Table 97. SAVEPRO output data element requirements.

Name	Description	Units
<i>grad</i>	2-dimensional array containing gradients of each profile used in XO calculations	M-units /meter
<i>htr</i>	2-dimensional array containing heights of each profile used in XO calculations	meters
<i>lvl</i>	Number of height levels in each profile used in XO calculations	N/A

5.2.18 Spectral Estimation (SPECEST) SU

The purpose of the SPECEST SU is to determine the outward propagation angle at the top of the PE calculation region or the grazing angle at the lower part of the PE region based on spectral estimation. The outward propagation angle is used for XO calculations and the grazing angle is used for rough surface calculations.

Upon entering the SPECEST SU, if the outward propagation angle is to be determined ($i_{flag} = 0$), the topmost n_p points (within the unfiltered portion) of the complex PE field are separated into their real and imaginary components, *xp* and *yp*, respectively. If the grazing angle is to be determined ($i_{flag} = 1$), then the lowest n_p points of the complex PE field are used. A window filter is then applied to both real and imaginary component arrays by multiplying each element in *xp* and *yp* by each corresponding element in the filter array *filtp* for indices between $\frac{3}{4} n_p$ and n_p .

Next, the array elements in xp and yp are set to 0 for indices from n_p+1 to n_s-1 . [Note that both xp and yp are arrays of size n_s .] The DRST SU is then referenced to obtain the spectral field components.

The spectral amplitudes in dB are then given by

$$spectr_i = 10 \text{LOG}_{10} \left[\text{MAX} \left(10^{-10}, \sqrt{xp_i^2 + yp_i^2} \right) \right]; \quad i = 0, 1, 2, \dots, n_s - 1.$$

Next, a 3-point average is performed on $spectr$ to determine the bin, or index i_{peak} , at which the peak spectral amplitude occurs. Once i_{peak} has been determined, the outward propagation angle is calculated as

$$\vartheta_{out} = \text{SIN}^{-1} \left(\frac{\lambda i_{peak}}{2 n_s \Delta z_{PE}} \right).$$

Table 98 and Table 99 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the SPECEST SU.

Table 98. SPECEST SU input data element requirements.

Name	Description	Units	Source
Δz_{PE}	PE mesh height increment (bin width in z-space)	meters	FFTPAR SU
$filt_p$	Array filter for spectral estimation calculations	N/A	APMINIT CSC
i_{flag}	Flag indicating if spectral estimation is to be performed on lower PE field or upper PE field 0 = upper PE field 1 = lower PE field	N/A	Calling SU
jz_{lim}	PE bin # corresponding to z_{lim} , i.e., $z_{lim} = jz_{lim} \Delta z_{PE}$	N/A	APMINIT CSC
ln_p	Power of 2 transform size used in spectral estimation calculations; i.e., $n_p = 2^{ln_p}$	N/A	APMINIT CSC
np_{34}	$\frac{3}{4} n_p$	N/A	APMINIT CSC
n_p	Number of bins in upper PE region to consider for spectral estimation.	N/A	APMINIT CSC
n_s	Transform size for spectral estimation calculations	N/A	APMINIT CSC
U	Complex field at current PE range r	$\mu\text{V/m}$	PESTEP SU
xo_{con}	Constant used in determining ϑ_{out}	N/A	APMINIT CSC
y_{cur}	Height of ground at current range r	meters	PESTEP SU

Table 99. SPECEST output data element requirements.

Name	Description	Units
<i>spectr</i>	Spectral amplitude of field	dB
<i>ϑ_{out}</i>	Outward propagation angle at top of PE region	radians
<i>x_p</i>	Real part of spectral portion of PE field	μV/m
<i>y_p</i>	Imaginary part of spectral portion field	μV/m

5.2.19 Troposcatter (TROPOSCAT) SU

The purpose of the TROPOSCAT SU is to calculate loss due to troposcatter and determine the appropriate loss to add to the already calculated propagation loss at and beyond the radio horizon.

Upon entering the TROPOSCAT SU, the current output range r_{out} is updated, and the surface refractivity and associated variables are also initialized. The surface refractivity sn_{ref} is initialized accordingly:

$$sn_{ref} = \frac{1}{2}(sn_{ref_{tx}} + ref_{ref_0}).$$

A term used in the troposcatter transmission loss calculation, sn_1 , is determined from

$$sn_1 = 0.031 - 0.00232 sn_{ref} + 5.67 \times 10^{-6} sn_{ref}^2,$$

along with a loss term $tlst_s$ for smooth surface as

$$tlst_s = tlst_{wr} - .2sn_{ref}.$$

Next, the tangent angle from the source to the surface, ϑ_1 , is initialized to its value for smooth surface, ϑ_{1s} ; and the troposcatter loss term $tlst$ is initialized to its value for smooth surface, $tlst_s$.

If performing a terrain case ($f_{ter} = \text{'true.'}$), the index j_{t2} is initialized where the first occurrence of the condition $j_{t2}\Delta r_{PE} > r_{out}$ is met. The index j_{t1} is set equal to the index location within $\vartheta_1 t$, up to $j_{t2}-1$, where the minimum value occurs. The tangent angle from the source height ϑ_1 is then initialized to $\vartheta_1 t_{j_{t1}}$ and the corresponding range d_1 is initialized $j_{t1}\Delta r_{PE}$.

The following steps 1 through 13 are performed for each output height index j from j_s to j_e .

1. If running a smooth surface case ($f_{ter} = \text{'false.'}$) and r_{out} is less than the minimum range rdt_j at which diffraction field solutions are applicable for the current height, then the SU is exited. Otherwise, the SU program flow continues with step 2.
2. The tangent angle from the receiver, ϑ_2 , is initialized to that for the j^{th} receiver height over smooth surface, ϑ_{2s_j} . However, if $f_{ter} = \text{'true.'}$, then the largest tangent angle a_2 and range d_2 from the receiver to the tangent point are determined using an iterative loop performed for index i from $j_{l2}-1$ to j_{l1} in decrements of -1 as follows:

$$r_2 = r_{out} - i \Delta r_{PE}$$

$$a_2 = \frac{z_{out_j} - tyh_i}{r_2} + \frac{r_2}{twoka},$$

If the current ϑ_2 value is less than a_2 , then ϑ_2 is set equal to a_2 and d_2 is set equal to r_2 . The index i is decremented by one, and the above calculations are repeated.

3. Once the loop in step 2 is been completed, the final value of ϑ_2 is checked. If it is greater than the tangent angle for smooth surface ϑ_{2s_j} (at the same receiver range), then both ϑ_2 and d_2 are set equal to ϑ_{2s_j} and d_{2s_j} , respectively.
4. Next, if r_{out} is less than the sum of the tangent ranges, d_1 and d_2 , then the SU is exited. Otherwise, SU program flow continues with step 5.
5. To account for antenna pattern effects over terrain, the ANTPAT SU is referenced using the tangent angle from the source to determine the antenna pattern factor, $f(\vartheta_1)$. The troposcatter loss term is then adjusted from its smooth surface value as

$$tlst = tlst_s - 20 \text{LOG}_{10}[f(\vartheta_1)].$$

The following steps 6 through 13 are now performed regardless of the value of f_{ter} .

6. The common volume scattering angle is given by

$$\theta = \vartheta_{0_{istp}} - \vartheta_1 - \vartheta_2.$$

7. Next, the following calculations are made to determine the effective scattering height h_o :

$$\begin{aligned}
a &= \frac{1}{2} \vartheta 0_{istp} - \vartheta_1 + \frac{adif_j}{r_{out}}, \\
b &= \frac{1}{2} \vartheta 0_{istp} - \vartheta_2 - \frac{adif_j}{r_{out}}, \\
s &= \mathbf{MIN} \left(\mathbf{MAX} \left(0.1, \frac{a}{b} \right), 10 \right), \\
h_o &= \frac{s r_{out} \theta}{10^3 (1 + s^2)}.
\end{aligned}$$

8. The parameter η_s is then calculated as a function of h_o :

$$\begin{aligned}
\eta_{sx} &= .5696 h_o \left(1 + s n_1 e^{-3.8 \times 10^{-6} h_o^6} \right), \\
\eta_s &= \mathbf{MIN}(\mathbf{MAX}(.01, \eta_{sx}), 5).
\end{aligned}$$

9. Next, the parameters ct_1 and ct_2 are defined as

$$\begin{aligned}
ct_1 &= 16.3 + 13.3 \eta_s \\
ct_2 &= .4 + .16 \eta_s,
\end{aligned}$$

where these are in turn used to calculate the quantities r_1 and r_2 :

$$\begin{aligned}
r_1 &= \mathbf{MAX}(0.1, r_t \theta), \\
r_2 &= \mathbf{MAX}(0.1, r_f zout_j \theta).
\end{aligned}$$

The quantity r_f was previously determined by referencing the TROPOINT SU.

10. ct_1 , ct_2 , r_1 , and r_2 are next used to determine H_1 and H_2 ;

$$\begin{aligned}
H_1 &= \mathbf{MAX} \left[0, ct_1 (r_1 + ct_2)^{-4/3} \right], \\
H_2 &= \mathbf{MAX} \left[0, ct_1 (r_2 + ct_2)^{-4/3} \right].
\end{aligned}$$

11. The frequency gain function H_o is then determined by

$$H_o = \frac{H_1 + H_2}{2} + \Delta H_o$$

where

$$\Delta H_o = 6[.6 - \text{LOG}_{10}(\eta_s)] \text{LOG}_{10}(s) \text{LOG}_{10}(q_t),$$

$$q_t = \text{MIN} \left[10, \text{MAX} \left(0.1, \frac{r_2}{s r_1} \right) \right].$$

ΔH_o is not allowed to be larger than $\frac{1}{2}(H_1 + H_2)$ and H_o is set equal to 0 if it becomes negative.

12. Next, the troposcatter loss is computed from

$$t_{loss} = t_{lst} + 573\theta + r \log o_{i_{stp}} + H_o.$$

13. Finally, through a method of “bold interpolation” the propagation loss $rloss$, previously computed in the CALCLOS SU, is adjusted for loss due to troposcatter according to

$$L_{dif} = rloss_j - t_{loss},$$

$$rloss_j = t_{loss} \quad \text{for } L_{dif} \geq 18 \text{ dB},$$

$$rloss_j = rloss_j - 10 \text{LOG}_{10} \left(1 + 10^{0.1 L_{dif}} \right) \quad \text{for } L_{dif} \geq -18 \text{ dB}.$$

Table 100 and Table 101 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the TROPOSCAT SU.

Table 100. TROPOSCAT SU input data element requirements.

Name	Description	Units	Source
$adif$	Height differences between ant_{ref} and all output receiver heights	meters	TROPOINT SU
$d2s$	Array of tangent ranges for all output receiver heights over smooth surface	meters	TROPOINT SU
e_k	Effective earth's radius factor	N/A	GET_K SU
f_{ter}	Logical flag indicating if terrain profile has been specified: ‘.true.’ = terrain profile specified ‘.false.’ = terrain profile not specified	N/A	TERINIT SU
i_{PE}	Number of PE range steps	N/A	PEINIT SU
i_{stp}	Current output range step index	N/A	Calling SU
j_e	Ending receiver height index at which to compute troposcatter loss	N/A	Calling SU
j_s	Starting receiver height index at which to compute troposcatter loss	N/A	Calling SU

Table 100. TROPOSCAT SU input data element requirements. (Continued)

Name	Description	Units	Source
j_{t2}	Index counter for tyh array indicating location of receiver range	N/A	TROPOINT SU APMSTEP CSC
rdt	Array of minimum ranges at which diffraction field solutions are applicable (for smooth surface) for all output receiver heights.	meters	TROPOINT SU
r_f	Constant used for troposcatter calculations	meters ⁻¹	TROPOINT SU
$rlogo$	Array containing 20 times the logarithm of all output ranges	N/A	APMINIT CSC
$rloss$	Propagation loss array	dB	Calling SU
rt_1	$r_f * ant_{ref}$	N/A	TROPOINT SU
$rngout$	Array containing all desired output ranges	meters	APMINIT CSC
$snref_{tx}$	Surface refractivity at transmitter	M-unit	REFINIT SU
ϑ_0	Array of angles used to determine common volume scattering angle	radians	TROPOINT SU
ϑ_{1s}	Tangent angle from source (for smooth surface)	radians	TROPOINT SU
ϑ_{2s}	Array of tangent angles from all output receiver heights - used with smooth surface	radians	TROPOINT SU
ϑ_{1t}	Array of tangent angles from source height - used with terrain profile	radians	TROPOINT SU
$tlst_{wr}$	Troposcatter loss term	dB	TROPOINT SU
$twoka$	Twice the effective earth radius	meters	GET_K SU
tyh	Adjusted height points of terrain profile at every PE range step.	meters	PEINIT SU
$zout$	Array containing all desired output heights referenced to h_{minter}	meters	APMINIT CSC

Table 101. TROPOSCAT SU output data element requirements.

Name	Description	Units
$rloss$	Propagation loss array	dB

5.3 EXTENDED OPTICS INITIALIZATION (XOINIT) CSC

The purpose of the XOINIT CSC is to initialize the range, height, and angle arrays in preparation for XO calculations.

Upon entering the XOINIT CSC, the value of i_{xostp} is tested. If i_{xostp} is equal to 0, then the APMCLEAR SU is referenced to deallocate all arrays used in the APM application and the CSC is exited. If i_{xostp} is greater than 0, then the following procedure is performed.

The arrays *curang* and *curng*, used for storage of traced local angles and ranges, respectively, are allocated and initialized to the range and angle values stored in *ffacz*. The array *curht* is allocated and initialized to the height of the top of the PE calculation region, z_{lim} . The array *igrd*, used for storage of starting refractivity gradient level (at which to begin ray tracing), is allocated and initialized to 0. The 2-dimensional array *prfhxo*, containing final output heights and propagation factors, along with the dummy array *dum*, used for temporary storage, are also allocated and initialized to 0.

If f_{ter} is ‘.true.’, then the MEANFILT SU is referenced twice to perform a 9-point smoothing operation on the angle values, using *dum* for temporary storage of angles after the first pass smoothing operation. Next, the starting height index at which to begin XO calculations, j_{xstart} is initialized to the ending height index for PE calculations, j_{end} , plus one. Finally, *dum* is deallocated before exiting.

Table 102 and Table 103 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the XOINIT CSC.

Table 102. XOINIT CSC input data element requirements.

Name	Description	Units	Source
<i>ffacz</i>	Array containing propagation factor, range, and propagation angle at z_{lim}	dB, meters, radians	FZLIM SU
f_{ter}	Logical flag indicating if terrain profile has been specified: .true. = terrain profile specified .false. = terrain profile not specified	N/A	TERINIT SU
i_{xostp}	Current output range step index for XO calculations	N/A	Calling SU
<i>iz</i>	Number of propagation factor, range, angle triplets stored in <i>ffacz</i>	N/A	FZLIM SU APMINIT CSC
iz_{max}	Maximum number of points allocated for arrays associated with XO calculations	N/A	APMINIT CSC
j_{end}	Ending index within <i>mpfl</i> of PE loss values	N/A	Calling SU
z_{lim}	Height limit for PE calculation region	meters	GETTHMAX SU

Table 103. XOINIT CSC output data element requirements.

Name	Description	Units
<i>curang</i>	Array of current local angles for each ray being traced in XO region	radians
<i>curht</i>	Array of current local heights for each ray being traced in XO region	meters
<i>curng</i>	Array of current local ranges for each ray being traced in XO region	meters
<i>i_{error}</i>	Error flag	N/A
<i>igrd</i>	Integer indexes indicating at what refractive gradient level to begin ray tracing for next XO range step for each ray in XO region.	N/A
<i>j_{xstart}</i>	Starting index within <i>mpfl</i> of XO loss values	N/A
<i>prfhxo</i>	2-dimensional array of propagation factor and heights for each ray traced in XO region to range <i>r_{out}</i>	dB,meters

5.3.1 APM Clean (APMCLEAN) SU

The purpose of the APMCLEAN SU is to deallocate all dynamically dimensioned arrays used in one complete run of APM calculations.

Upon entry, all arrays that were dynamically allocated at the beginning of the current application are now deallocated.

Table 104 and Table 105 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the APMCLEAN SU.

Table 104. APMCLEAN CSC input data element requirements.

Name	Description	Units	Source
<i>adif</i>	Height array used for troposcatter calculations	meters	TROPOINT SU
<i>curang</i>	Array of current local angles for each ray being traced in XO region	radians	EXTO SU XOINIT CSC
<i>curht</i>	Array of current local heights for each ray being traced in XO region	meters	EXTO SU XOINIT CSC
<i>curng</i>	Array of current local ranges for each ray being traced in XO region	meters	EXTO SU XOINIT CSC
<i>d2s</i>	Array of tangent ranges for all output receiver heights over smooth surface	meters	TROPOINT SU
<i>dielec</i>	2-dimensional array containing the relative permittivity and conductivity; <i>dielec_{1,i}</i> and <i>dielec_{2,i}</i> , respectively.	N/A, S/m	Calling CSCI, DIEINIT SU
<i>envpr</i>	Complex [refractivity] phase term array interpolated every ΔZ_{PE} in height	N/A	PEINIT SU PESTEP SU

Table 104. APMCLEAR CSC input data element requirements. (Continued)

Name	Description	Units	Source
<i>ffacz</i>	Array containing propagation factor, range, and propagation angle at z_{lim}	dB, meters, radians	FZLIM SU
<i>ffrout</i>	Array of propagation factors at each output range beyond r_{atz} and at height z_{lim}	dB	CALCLOS SU
<i>filt</i>	Cosine-tapered (Tukey) filter array	N/A	PEINIT SU
<i>filtp</i>	Array filter for spectral estimation calculations	N/A	APMINIT CSC
<i>frsp</i>	Complex free-space propagator term array	N/A	PEINIT SU
<i>fsl</i>	Free space loss array for output ranges	dB	APMINIT CSC
<i>gr</i>	Intermediate M-unit gradient array, RO region	(M-unit/m) 10^{-6}	REFINIT SU
<i>grad</i>	2-dimensional array containing gradients of each profile used in XO calculations	M-units /meter	SAVEPRO SU
<i>grdum</i>	Array of refractivity gradients defined by profile <i>htdum</i> and <i>refdum</i>	M-units /meter	REFINIT SU REFINTER SU
<i>hfangr</i>	Array of user-defined cut-back angles. This is used only for user-defined height-finder antenna type.	radians	APMINIT CSC
<i>hlim</i>	Array containing height at each output range separating the RO region from the PE (at close ranges) and XO (at far ranges) regions	meters	GETTHMAX SU
<i>href</i>	Heights of refractivity profile with respect to y_{ref}	meters	PROFINT SU
<i>ht</i>	PE mesh height array of size n_{ft}	meters	PEINIT SU
<i>htdum</i>	Height array for current interpolated profile	meters	REFINIT SU REFINTER SU
<i>htfe</i>	Array containing the height at each output range separating the FE region from the RO region (full hybrid mode), or the FE region from the PE region (partial hybrid mode)	meters	FILLHT SU
<i>htr</i>	2-dimensional array containing heights of each profile used in XO calculations	meters	SAVEPRO SU
<i>igrd</i>	Integer indexes indicating at what refractive gradient level to begin ray tracing for next XO range step for each ray in XO region.	N/A	XOINIT CSC

Table 104. APMCLEAR CSC input data element requirements. (Continued)

Name	Description	Units	Source
<i>igrnd</i>	Integer array containing ground type composition for given terrain profile - can vary with range. Different ground types are: 0 = sea water 1 = fresh water 2 = wet ground 3 = medium dry ground 4 = very dry ground 5 = ice at -1 degree C 6 = ice at -10 degree C 7 = user defined (in which case, values of relative permittivity and conductivity must be given).	N/A	Calling CSCI
<i>lvl</i>	Number of height levels in each profile used in XO calculations	N/A	SAVEPRO SU
<i>nc²</i>	Array of complex dielectric constants	N/A	DIEINIT SU
<i>prfhxo</i>	Array of propagation factor and heights for each ray traced in XO region to range r_{out}	dB,meters	XOINIT CSC
<i>profint</i>	Profile interpolated to every Δz_{PE} in height	M-units	REFINTER SU
ψ	Array of interpolated grazing angles at each PE range step	radians	GRAZE_INT SU
<i>q</i>	Intermediate M-unit difference array, RO region	2M-unit 10^{-6}	REFINIT SU
<i>rdt</i>	Array of minimum ranges at which diffraction field solutions are applicable (for smooth surface) for all output receiver heights.	meters	TROPOINT SU
<i>refdum</i>	M-unit array for current interpolated profile	M-units	REFINIT SU REFINTER SU
<i>refref</i>	Refractivity profile with respect to y_{ref}	M-units	PROFINT SU
<i>rfac1</i>	Propagation factor at valid output height points from PE field at range r_{last}	dB	CALCLOS SU
<i>rfac2</i>	Propagation factor at valid output height points from PE field at range r	dB	CALCLOS SU
<i>rgrnd</i>	Array containing ranges at which varying ground types apply.	meters	Calling CSCI
<i>rlogo</i>	Array containing 20 times the logarithm of all output ranges	N/A	APMINIT CSC
<i>rloss</i>	Propagation loss	dB	ALLARRAY_APM CALCLOS SU EXTO SU TROPOSCAT SU
<i>rm</i>	Intermediate M-unit array, RO region	M-unit 10^{-6}	REFINIT SU
<i>rngout</i>	Array containing all desired output ranges	meters	APMINIT CSC
<i>rn</i>	Array of R_T to the i^{th} power (e.g., $m_i = R_T^i$)	N/A	GETALN SU

Table 104. APMCLEAR CSC input data element requirements. (Continued)

Name	Description	Units	Source
<i>rsqrd</i>	Array containing the square of all desired output ranges	meters ²	APMINIT CSC
<i>spectr</i>	Spectral amplitude of field	dB	SPECEST SU
<i>v0</i>	Array of angles used to determine common volume scattering angle	radians	TROPOINT SU
<i>v2s</i>	Array of tangent angles from all output receiver heights - used with smooth surface	radians	TROPOINT SU
<i>v1t</i>	Array of tangent angles from source height - used with terrain profile	radians	TROPOINT SU
<i>tyh</i>	Adjusted height points of terrain profile	meters	PEINIT SU
<i>U</i>	Complex PE field	μV/m	PESTEP SU
<i>udum</i>	Real or imaginary part of complex field array	μV/m	FFT SU
<i>Ulast</i>	Complex PE field at range r_{last}	μV/m	PESTEP SU
<i>w</i>	Difference equation of complex PE field	μV/m ²	PESTEP SU
<i>xp</i>	Real part of spectral portion of PE field	μV/m	SPECEST SU
<i>ym</i>	Particular solution of difference equation	μV/m	PESTEP SU
<i>yp</i>	Imaginary part of spectral portion field	μV/m	SPECEST SU
<i>zout</i>	Array containing all desired output heights referenced to h_{minter}	meters	APMINIT CSC
<i>zoutma</i>	Output height point relative to “real” ant_{ref}	meters	APMINIT CSC
<i>zoutpa</i>	Output height point relative to “image” ant_{ref}	meters	APMINIT CSC
<i>zRO</i>	Array of output heights in RO region	meters	APMINIT CSC
<i>zrt</i>	Intermediate height array, RO region	meters	REFINIT SU

Table 105. APMCLEAR CSC output data element requirements.

Name	Description	Units
<i>i_{error}</i>	Error flag indicator: non-zero if error has occurred in deallocation procedure	N/A

5.3.2 Mean Filter (MEANFILT) SU

The purpose of the MEANFILT SU is to perform a i_{sz} -point average smoothing operation on the array passed to it.

The array *arbef* is passed to the SU, along with the number of points over which to perform the smoothing operation, i_{sz} . Once the smoothing operation has been performed, the resulting “smoothed” points are stored in *arapt* and passed back to the calling routine. The operation is performed as follows:

$$araft_k = \frac{1}{m} \sum_{i=k-m'}^{k+m'} arbef_i \quad \text{for } k = m' + 1, m' + 2, \dots, i_{sz} - m'$$

where m' is $\frac{1}{2}(m-1)$ and m is the size of array *arbef*.

Table 106 and Table 107 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the MEANFILT SU.

Table 106. MEANFILT SU input data element requirements.

Name	Description	Units	Source
<i>arbef</i>	Array of angles before smoothing operation	radians	Calling SU
<i>i_{sz}</i>	Number of points over which to perform average smoothing	N/A	Calling SU
<i>m</i>	Size of array <i>arbef</i>	N/A	Calling SU

Table 107. MEANFILT SU output data element requirements.

Name	Description	Units
<i>araft</i>	Array of angles after smoothing operation	radians

5.4 EXTENDED OPTICS STEP (XOSTEP) CSC

The purpose of the XOSTEP CSC is to calculate the propagation loss in the XO region for one output range step.

Upon entering the XOSTEP CSC, the current execution mode is checked to determine if XO calculations will be necessary ($i_{hybrid} \neq 0$). If i_{hybrid} is 0, then the CSC is exited.

If i_{hybrid} is not equal to 0, the output range r_{out} , and the gaseous absorption loss gas_{loss} are updated. The *mpfl* values are initialized to -1000 from the index of the start of XO calculations, j_{xstart} , to the maximum number of height output points, n_{zout} . The EXTO SU is then referenced to calculate propagation factor and loss values in the XO region. Propagation factor and loss values are returned in *mpfl* from index j_{xstart} to j_{xe} .

If FE and RO calculations need to be performed ($i_{hybrid} = 1$), then the indices j_{fs} and j_{fe} , indicating the height index at which to start and end FE calculations, respectively, are determined. The FEM SU is then referenced to compute propagation factor and loss

values for heights $zout_{j_{fs}}$ to $zout_{j_{fe}}$. Similarly for RO calculations, the indices j_{rs} and j_{re} are determined, and the ROLOSS SU is referenced to compute propagation factor and loss values for heights $zout_{j_{rs}}$ to $zout_{j_{re}}$.

Finally, the index j_{xend} is set equal to the maximum of j_{xe} , j_{fe} , and j_{re} . If the last range value has been reached ($i_{stp} = n_{rout}$) then the APMCLEAR SU is referenced to deallocate all arrays allocated for the APM application.

Table 108 and Table 109 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the XOSTEP CSC.

Table 108. XOSTEP CSC input data element requirements.

Name	Description	Units	Source
gas_{att}	Gaseous absorption attenuation rate	dB/km	GASABS SU
ht_{fe}	Array containing the height at each output range separating the FE region from the RO region (full hybrid mode), or the FE region from the PE region (partial hybrid mode)	meters	FILLHT SU
ht_{jim}	Maximum height relative to h_{minter}	meters	TERINIT SU
i_{hybrid}	Integer indicating which sub-models will be used: 0 = pure PE model 1 = full hybrid model (PE + FE + RO + XO) 2 = partial hybrid model (PE + XO)	N/A	APMINIT CSC
i_{stp}	Current output range step index	N/A	Calling CSCI
j_{xstart}	Index at which valid propagation factor and loss values in $mpfl$ start	N/A	Calling CSCI
n_{zout}	Integer number of output height points desired	N/A	Calling CSCI
$rngout$	Array containing all desired output ranges	meters	APMINIT CSC
$zout$	Array containing all desired output heights referenced to h_{minter}	meters	APMINIT CSC

Table 109. XOSTEP CSC output data element requirements.

Name	Description	Units
j_{xend}	Index at which valid propagation factor and loss values in $mpfl$ end	N/A
$mpfl$	Propagation factor and loss array	cB
r_{out}	Current desired output range	meters

5.4.1 Extended Optics (EXTO) SU

The purpose of the EXTO SU is to calculate propagation factor and loss based on XO techniques. The SU performs a ray trace on all rays within one output range step and returns the propagation factor and loss up to the necessary height; storing all angle, height, and range information for subsequent ray tracing upon the next reference to the SU.

Upon entering the SU, internal one-line ray trace functions are defined as

$$\mathbf{RADA1}(a,b)=a^2+2g_{rd}b,$$

$$\mathbf{RP}(a,b)=a+\frac{b}{g_{rd}},$$

$$\mathbf{AP}(a,b)=a+bg_{rd},$$

$$\mathbf{HP}(a,b,c)=a+\frac{b^2-c^2}{2g_{rd}}.$$

Next, the starting and ending index counters iz_s , iz_e , respectively, for the local angle, range, and height arrays, and the refractivity profile starting index i_{rps} are initialized to 1 for the first reference to the EXTO SU. The index iz_e is then determined such that $curng_{iz_e} \leq r_{out} < curng_{iz_e+1}$. The integer counter k , indicating the number of propagation factor and heights in array *prfhxo*, is initialized to 0.

The following ray trace steps 1 through 3 are performed for each ray; i.e., for each j^{th} angle, range, and height triplet, for j ranging from iz_s to iz_e .

1. At the start of the ray trace, the current local angle (a_0), range (r_0), height (h_0), and refractive gradient index (i_{grad}) are initialized to $curang_j$, $curng_j$, $curht_j$, and $igrd_j$, respectively. Next, refractive profile index, i_{rp} , is initialized to the maximum of j or i_{rps} . Finally, the refractivity gradient, g_{rd} is set equal to the gradient at the i_{grad}^{th} level of the i_{rp}^{th} profile, $grad_{i_{grad} \cdot i_{rp}}$. The following ray trace steps 1.a through 1.d are then performed until the current local range r_0 becomes greater than or equal to r_{out} .
 - a. The ending range, r_1 , in the ray trace segment is set equal to the minimum of $ffacz_{i_{rp}+1,2}$ or r_{out} . If i_{rp} is equal to the number of stored triplets, iz , then r_1 is set equal to r_{out} .
 - b. The j^{th} ray is then traced to r_1 and the resulting angle and height at the end of the segment is determined via the in-line functions as

$$a_1 = \mathbf{AP}(a_0, r_1 - r_0),$$

$$h_1 = \mathbf{HP}(h_0, a_1, a_0).$$

- c. The ending height h_1 is then compared to the next height level in the current refractivity profile, $htr_{i_{grad}+1, i_{rp}}$ and if h_1 is greater than this height level, it is set equal to $htr_{i_{grad}+1, i_{rp}}$ and a new a_1 and r_1 are computed from

$$\begin{aligned} a_1 &= \sqrt{\text{RADA1}(a_0, h_1 - h_0)} \\ r_1 &= \text{RP}(r_0, a_1 - a_0) \end{aligned}$$

i_{grad} is then set to the minimum of $i_{grad}+1$ or $lvl_{i_{rp}}-1$.

- d. The starting angle, range, and height for the next ray trace segment is updated, and, if necessary, the refractivity profile index i_{rp} is updated to the minimum of $i_{rp}+1$ or iz_e . Steps 1.a through 1.d are then repeated for the next ray segment.
2. Once the ray has been traced to a range of r_{out} or greater, the current angle, range, and height arrays, $curang$, $curng$, and $curht$, respectively, are updated to the values for a_0 , r_0 , and h_0 for subsequent references to the EXTO SU.
3. The counter k for the propagation factor and height array is incremented by one and the array is updated according to

$$\begin{aligned} prfho_{k,1} &= ffacz_{j,1}, \\ prfho_{k,2} &= h_0. \end{aligned}$$

Once all rays have been traced, the starting profile index i_{rps} is updated to iz_e for the next reference to the EXTO SU, and the counter k is again incremented by one and the last values of $prfho$ updated as follows,

$$\begin{aligned} prfho_{k,1} &= ffrout_{i_{stp},1}, \\ prfho_{k,2} &= ffrout_{i_{stp},2}. \end{aligned}$$

The number of traced XO height points, n_{xo} , at the current output range is then set to k . Note that at this point, all output heights in $prfho_{0:n_{xo},2}$ are decreasing from $prfho_{1,2}$ to $prfho_{n_{xo},2}$ and all traced heights in $curht$ are decreasing from $curht_{iz_s}$ to $curht_{iz_e}$.

The starting index iz_s is then adjusted (for the next reference to the EXTO SU) if the topmost traced height $curht_{iz_s}$ is greater than ht_{lim} . If performing a terrain case, the output height points may not be continually decreasing from $prfho_{1,2}$ to $prfho_{n_{xo},2}$. In this case, $prfho$ is sorted such that all height values are steadily decreasing. The ending index, j_{xe} , at which XO propagation factor and loss values will be calculated and stored in

$mpfl$, is set equal to n_{zout} and adjusted, if necessary, such that $zout_{j_{xe}}$ is less than $prfhxo_{1,2}$. Now, the counter index ix is initialized to n_{xo} . Next, the propagation factor is determined via linear interpolation on the values in $prfhxo$. The following steps 1 through 2 are performed for each output height point $zout_j$ for j varying from j_{xs} to j_{xe} .

1. The counter ix is adjusted (if necessary) such that $prfhxo_{ix,2} \leq zout_j < prfhxo_{ix+1,2}$.
2. The propagation factor F_{dB} and propagation loss $rloss$ at height $zout_j$ are then calculated according to

$$F_{dB} = prfhxo_{ix-1,1} + (prfhxo_{ix,1} - prfhxo_{ix-1,1}) \frac{zout_j - prfhxo_{ix,2}}{prfhxo_{ix-1,2} - prfhxo_{ix,2}},$$

$$rloss_j = fsl_{istp} - F_{dB}$$

Once all propagation loss values have been computed, the TROPOSCAT SU is referenced to compute troposcatter loss, if necessary. Finally, the loss due to gaseous absorption is added to $rloss$ and then converted to centibels and stored in $mpfl$ before exiting.

Table 110 and Table 111 identify, describe the purpose for, state the units of, and show the computational source for each input and output data element, respectively, of the EXTO SU. Table 112 identifies terms which are used internal to the EXTO SU and whose value must be retained from SU call to SU call for reasons of computational efficiency.

Table 110. EXTO SU input data element requirements.

Name	Description	Units	Source
<i>curang</i>	Array of current local angles for each ray being traced in XO region	radians	EXTO SU XOINIT CSC
<i>curht</i>	Array of current local heights for each ray being traced in XO region	meters	EXTO SU XOINIT CSC
<i>curng</i>	Array of current local ranges for each ray being traced in XO region	meters	EXTO SU XOINIT CSC
<i>ffacz</i>	Array containing propagation factor, range, and propagation angle at z_{lim}	dB, meters, radians	FZLIM SU
<i>ffrout</i>	Array of propagation factors at each output range beyond r_{atz} and at height z_{lim}	dB	CALCLOS SU
<i>fsl</i>	Free space loss array for output ranges	dB	APMINIT CSC

Table 110. EXTO SU input data element requirements. (Continued)

Name	Description	Units	Source
f_{ter}	Logical flag indicating if terrain profile has been specified: .true. = terrain profile specified .false. = terrain profile not specified	N/A	TERINIT SU
gas_{loss}	Gaseous absorption loss at range r_{out}	dB	APMSTEP CSC
$grad$	2-dimensional array containing gradients of each profile used in XO calculations	M-units /meter	SAVEPRO SU
$hlim$	Array containing height at each output range separating the RO region from the PE (at close ranges) and XO (at far ranges) regions	meters	GETTHMAX SU
ht_{lim}	Maximum height relative to h_{minter}	meters	TERINIT SU
htr	2-dimensional array containing heights of each profile used in XO calculations	meters	SAVEPRO SU
$igrd$	Integer indexes indicating at what refractive gradient level to begin ray tracing for next XO range step for each ray in XO region.	N/A	XOINIT CSC
i_{ratz}	Index of output range step in which to begin storing propagation factor and outgoing angle for XO region	N/A	APMINIT CSC
i_{stp}	Current output range step index	N/A	Calling SU
T_{ropo}	Troposcatter calculation flag: 'false.' = no troposcatter calcs 'true.' = troposcatter calcs	N/A	Calling CSCI
iz	Number of propagation factor, range, angle triplets stored in $ffacz$	N/A	FZLIM SU
j_{xs}	Index at which valid loss values in $mpfl$ start	N/A	Calling SU
lvl	Number of height levels in each profile used in XO calculations	N/A	SAVEPRO SU
n_{zout}	Integer number of output height points desired	N/A	Calling CSCI
r_{out}	Current output range	meters	Calling SU
$zout$	Array containing all desired output heights referenced to h_{minter}	meters	APMINIT CSC

Table 111. EXTO SU output data element requirements.

Name	Description	Units
<i>curang</i>	Array of current local angles for each ray being traced in XO region	radians
<i>curht</i>	Array of current local heights for each ray being traced in XO region	meters
<i>curng</i>	Array of current local ranges for each ray being traced in XO region	meters
<i>hlim</i>	Array containing height at each output range separating the RO region from the PE (at close ranges) and XO (at far ranges) regions	meters
<i>j_{xe}</i>	Index at which valid loss values in <i>mpfl</i> end	N/A
<i>mpfl</i>	2-dimensional propagation factor and loss array	cB
<i>prfhxo</i>	2-dimensional array of propagation factor and heights for each ray traced in XO region to range <i>r_{out}</i>	dB,meters
<i>rloss</i>	Propagation loss	dB

Table 112. EXTO SU save data element requirements.

Name	Description	Units
<i>i_{tps}</i>	Starting index counter for refractivity profiles	N/A
<i>iz_e</i>	Ending index in <i>curang</i> , <i>curng</i> , and <i>curht</i> to trace to <i>r_{out}</i>	N/A
<i>iz_s</i>	Starting index in <i>curang</i> , <i>curng</i> , and <i>curht</i> to trace to <i>r_{out}</i>	N/A

6. REQUIREMENTS TRACEABILITY

This section provides the traceability of the design of the APM CSCI. Table 113 presents this traceability between the corresponding sections of the Software Requirements Specification (SRS) and the Software Design Description (SDD) and between the various components of the APM CSCI.

Table 113. Traceability matrix between the SRS and the SDD.

Software Requirements Specification		Software Design Description	
SRS Requirement Name	SRS Paragraph Number	Software Design Description Name	SDD Paragraph Number
CSCI Capability Requirements	3.1	CSCI-WIDE DESIGN DECISIONS	3.
CSCI Capability Requirements	3.1	CSCI Components	4.1
CSCI Capability Requirements	3.1	Concept of Execution	4.2

Table 113 Traceability matrix between the SRS and the SDD. (Continued)

Software Requirements Specification		Software Design Description	
SRS Requirement Name	SRS Paragraph Number	Software Design Description Name	SDD Paragraph Number
Advance Propagation Initialization (APMINIT) CSC	3.1.1	Advance Propagation Initialization (APMINIT) CSC	5.1
Allocate Arrays APM (ALLARRAY_APM) SU	3.1.1.1	Allocate Arrays APM (ALLARRAY_APM) SU	5.1.1
Allocate Array PE (ALLARRAY_PE) SU	3.1.1.2	Allocate Array PE (ALLARRAY_PE) SU	5.1.2
Allocate Array RO (ALLARRAY_RO) SU	3.1.1.3	Allocate Array RO (ALLARRAY_RO) SU	5.1.3
Allocate Array (ALLARRAY_XORUF) SU	3.1.1.4	Allocate Array (ALLARRAY_XORUF) SU	5.1.4
Alpha Impedance Initialization (ALN_INIT) SU	3.1.1.5	Alpha Impedance Initialization (ALN_INIT) SU	5.1.5
Antenna Pattern (ANTPAT) SU	3.1.1.6	Antenna Pattern (ANTPAT) SU	5.1.6
Dielectric Initialization (DIEINIT) SU	3.1.1.7	Dielectric Initialization (DIEINIT) SU	5.1.7
FFT Parameters (FFTPAR) SU	3.1.1.8	FFT Parameters (FFTPAR) SU	5.1.8
Fill Height Arrays (FILLHT) SU	3.1.1.9	Fill Height Arrays (FILLHT) SU	5.1.9
Gaseous Absorption (GASABS) SU	3.1.1.10	Gaseous Absorption (GASABS) SU	5.1.10
Get Effective Earth Radius Factor (GET_K) SU	3.1.1.11	Get Effective Earth Radius Factor (GET_K) SU	5.1.11
Get Alpha Impedance (GETALN) SU	3.1.1.12	Get Alpha Impedance (GETALN) SU	5.1.12
Get Grazing Angle (GETGRAZE) SU	3.1.1.13	Get Grazing Angle (GETGRAZE) SU	5.1.13
Get Maximum Angle (GETTHMAX) SU	3.1.1.14	Get Maximum Angle (GETTHMAX) SU	5.1.14
Grazing Angle Interpolation (GRAZE_INT) SU	3.1.1.15	Grazing Angle Interpolation (GRAZE_INT) SU	5.1.15
Interpolate Profile (INTPROF) SU	3.1.1.16	Interpolate Profile (INTPROF) SU	5.1.16
PE Initialization (PEINIT) SU	3.1.1.17	PE Initialization (PEINIT) SU	5.1.17
Profile Reference (PROFREF) SU	3.1.1.18	Profile Reference (PROFREF) SU	5.1.18
RD Trace (RDTRACE) SU	3.1.1.19	RD Trace (RDTRACE) SU	5.1.19

Table 113 Traceability matrix between the SRS and the SDD. (Continued)

Software Requirements Specification		Software Design Description	
SRS Requirement Name	SRS Paragraph Number	Software Design Description Name	SDD Paragraph Number
Refractivity Initialization (RefInit) SU	3.1.1.20	Refractivity Initialization (REFINIT) SU	5.1.20
Remove Duplicate Refractivity Levels (REMDUP) SU	3.1.1.21	Remove Duplicate Refractivity Levels (REMDUP) SU	5.1.21
Terrain Initialization (TERINIT) SU	3.1.1.22	Terrain Initialization (TERINIT) SU	5.1.22
Trace to Output Range (TRACE_ROUT) SU	3.1.1.23	Trace to Output Range (TRACE_ROUT) SU	5.1.23
Troposcatter Initialization (TROPOINIT) SU	3.1.1.24	Troposcatter Initialization (TROPOINIT) SU	5.1.24
Starter Field Initialization (XYINIT) SU	3.1.1.25	Starter Field Initialization (XYINIT) SU	5.1.25
Advance Propagation Model Step (APMSTEP) CSC	3.1.2	Advance Propagation Model Step (APMSTEP) CSC	5.2
Airborne Hybrid Model (AIRBORNE) SU	3.1.2.1	Airborne Hybrid Model (AIRBORNE) SU	5.2.1
Calculate Propagation Loss (CALCLOS) SU	3.1.2.2	Calculate Propagation Loss (CALCLOS) SU	5.2.2
DOSHIFT SU	3.1.2.3	DOSHIFT SU	5.2.3
Discrete Sine/Cosine Fast-Fourier Transform (DRST) SU	3.1.2.4	Discrete Sine/Cosine Fast-Fourier Transform (DRST) SU	5.2.4
Flat-earth Model (FEM) SU	3.1.2.5	Flat-earth Model (FEM) SU	5.2.5
Fast-Fourier Transform (FFT) SU	3.1.2.6	Fast-Fourier Transform (FFT) SU	5.2.6
Free-Space Range Step (FRSTP) SU	3.1.2.7	Free-Space Range Step (FRSTP) SU	5.2.7
FZLIM SU	3.1.2.8	FZLIM SU	5.2.8
Get Propagation Factor (GETPFAC) SU	3.1.2.9	Get Propagation Factor (GETPFAC) SU	5.2.9
Get Reflection Coefficient (GETREFCOEF) SU	3.1.2.10	Get Reflection Coefficient (GETREFCOEF) SU	5.2.10
Mixed Fourier Transform (MIXEDFT) SU	3.1.2.11	Mixed Fourier Transform (MIXEDFT) SU	5.2.11
Parabolic Equation Step (PESTEP) SU	3.1.2.12	Parabolic Equation Step (PESTEP) SU	5.2.12

Table 113 Traceability matrix between the SRS and the SDD. (Continued)

Software Requirements Specification		Software Design Description	
SRS Requirement Name	SRS Paragraph Number	Software Design Description Name	SDD Paragraph Number
Ray Trace (RAYTRACE) SU	3.1.2.13	Ray Trace (RAYTRACE) SU	5.2.13
Refractivity Interpolation (REFINTER) SU	3.1.2.14	Refractivity Interpolation (REFINTER) SU	5.2.14
Ray Optics Calculation (ROCALC) SU	3.1.2.15	Ray Optics Calculation (ROCALC) SU	5.2.15
Ray Optics Loss (ROLOSS) SU	3.1.2.16	Ray Optics Loss (ROLOSS) SU	5.2.16
Save Profile (SAVEPRO) SU	3.1.2.17	Save Profile (SAVEPRO) SU	5.2.17
Spectral Estimation (SPECEST) SU	3.1.2.18	Spectral Estimation (SPECEST) SU	5.2.18
Troposcatter (TROPOSCAT) SU	3.1.2.17	Troposcatter (TROPOSCAT) SU	5.2.19
Extended Optics Initialization (XOINIT) CSC	3.1.3	Extended Optics Initialization (XOINIT) CSC	5.3
Advanced Propagation Model Clean (APMCLEAN) CSC	3.1.3.1	Advanced Propagation Model Clean (APMCLEAN) CSC	5.3.1
Mean Filter (MEANFILT) SU	3.1.3.2	Mean Filter (MEANFILT) SU	5.3.2
Extended Optics Step (XOSTEP) CSC	3.1.4	Extended Optics Step (XOSTEP) CSC	5.4
Extended Optics (EXTO) SU	3.1.4.1	Extended Optics (EXTO) SU	5.4.1
CSCI External Interface Requirements	3.2	External Interface	4.3.2
CSCI Internal Interface Requirements	3.3	Internal Interface	4.3.3
CSCI Internal Data Requirements	3.4	Internal Data	4.3.4
Environmental Radio Refractivity field Data Element	3.5.1	Environmental Radio Refractivity field Data Element	7.2
Terrain Profile Data Element	3.5.2	Terrain Profile Data Element	7.3
Implementation and Application Considerations	3.10.1	Implementation and Application Considerations	7.1

7. NOTES

7.1 APM CSCI IMPLEMENTATION AND APPLICATION CONSIDERATIONS

The calling NITES CSCI application will determine the employment of the APM CSCI. However, the intensive computational nature of the APM CSCI must be taken into consideration when designing an efficient calling application. For this reason, the APM CSCI is designed with flexibility for various hardware suites and computer resource management considerations. This APM CSCI applies only to a coverage and loss diagram application. The following highly recommended guidelines are provided to aid in the design of a coverage or loss diagram application which will most efficiently employ the APM CSCI.

The APM CSCI propagation loss calculations are independent of any target or receiver considerations; therefore, for any EM emitter, one execution of the APM CSCI may be used to create both a coverage diagram and a loss diagram. Since both execution time and computer memory allocation should be a consideration when employing this model, it is most efficient and appropriate to execute the APM CSCI for a particular EM system/environmental/terrain combination before executing any application. The output of the APM CSCI would be stored in a file which would be accessed by multiple applications.

For example, the NITES operator may desire a coverage diagram for one particular radar system. At the beginning of the coverage diagram application, a check would be made for the existence of a previously created APM CSCI output file appropriate for the EM system, environmental, and terrain conditions. If such a file exists, the propagation loss values would be read from the file and used to create the coverage diagram. If the file does not exist, the APM CSCI would be executed to create one. As the APM CSCI is executing, its output could be routed simultaneously to a graphics display device and a file. This file could then be used in the loss diagram application should the operator also choose it. Two distinct applications, therefore, are achieved with only one execution of the APM CSCI. Additionally, should the operator desire an individual coverage diagram for each of multiple targets, or a single coverage diagram illustrating radar detection of a low-flying missile superimposed upon a coverage diagram illustrating his own radar's vulnerability as defined by the missile's ESM receiver, only a single execution of the APM CSCI would be required, thereby saving valuable computer resources.

7.2 ENVIRONMENTAL RADIO REFRACTIVITY FIELD DATA ELEMENTS

The radio-refractivity field, i.e., the profiles of M-units versus height, should consist of vertical piece-wise linear profiles specified by couplets of height in meters with respect to mean sea level and modified refractivity (M-units) at multiple arbitrary ranges. All vertical profiles must contain the same number of vertical data points, and be

specified such that each numbered data point corresponds to like-numbered points (i.e., features) in the other profiles. The first numbered data point of each profile must correspond to a height of zero mean sea level and the last numbered data point must correspond to a height such that the modified refractivity for all greater heights is well represented by extrapolation using the two highest profile points specified.

With the inclusion of terrain and allowing the terrain profile to fall below mean sea level, refractivity profiles can also be provided in which the first level is less than 0 (or below mean sea level). For a terrain profile that falls below mean sea level at some point, the assumption is that the minimum height may be less than the first height in any refractivity profile specified. Therefore, an extrapolation flag, i_{extra} , must be specified to indicate how the APM CSCI should extrapolate from the first refractivity level to the minimum height along the terrain profile. Setting i_{extra} to 0 will cause the APM CSCI to extrapolate to the minimum height using a standard atmosphere gradient; setting i_{extra} to 1 will cause the APM CSCI to extrapolate to the minimum height using the gradient determined from the first two levels of the refractivity profile.

Within each profile, each numbered data point must correspond to a height greater than or equal to the height of the previous data point. Note that this requirement allows for a profile which contains redundant data points. Note also that all significant features of the refractivity profiles must be specified, even if they are above the maximum output height specified for a particular application of APM.

The NITES CSCI application designer and the NITES operator share responsibility for determining appropriate environmental inputs. For example, a loss diagram may be used to consider a surface-to-surface radar detection problem. Since the operator is interested in surface-to-surface, he may truncate the profile assuming that effects from elevated ducting conditions are negligible. It may be, however, that the elevated duct does indeed produce a significant effect. The operator should ensure, therefore, that the maximum height of the profile allows for the inclusion of all significant refractive features.

This specification allows a complicated refractivity field to be described with a minimum of data points. For example, a field in which a single trapping layer linearly descends with increasing range can be described with just two profiles containing only four data points each, frame (a) of Figure 3. In the same manner, other evolutions of refractive layers may be described. Frames (b) and (c) of Figure 3 show two possible scenarios for the development of a trapping layer. The scenario of choice is the one which is consistent with the true thermodynamical and hydrological layering of the atmosphere.

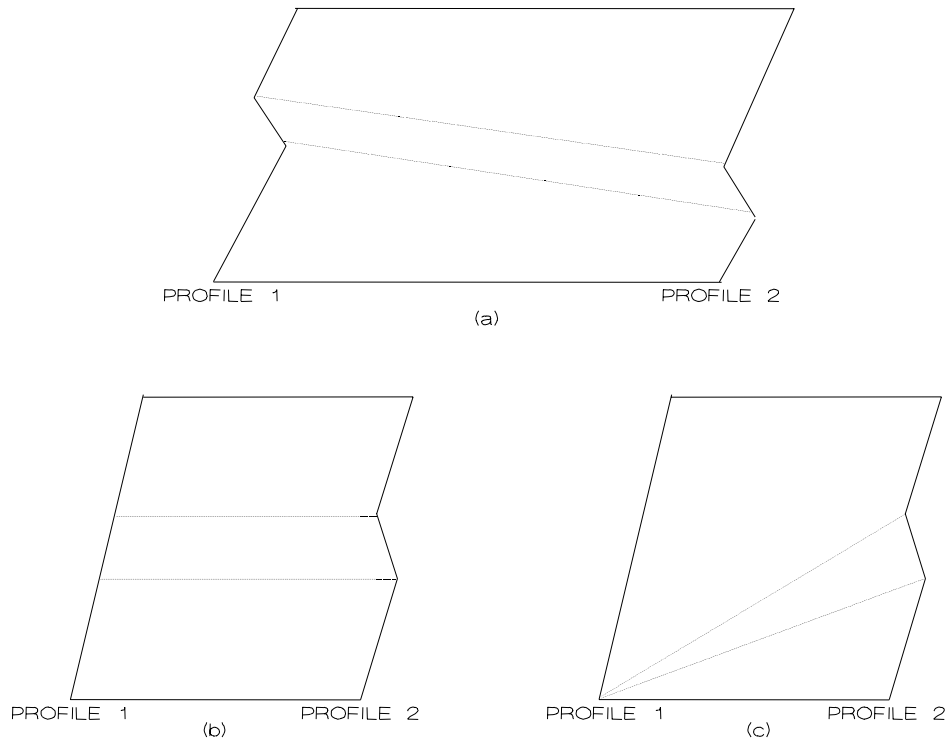


Figure 3. Idealized M-unit profiles (solid) and lines of interpolation (dashed).

Two external implementation data variables applicable to both the NITES operator and to the calling application designer are r_{max} , the maximum APM CSCI output range, and h_{max} , the maximum APM CSCI output height. These two parameters are required by the APM CSCI to determine the horizontal and vertical resolution, respectively, for internal range and height calculations based on the current values of n_{rout} and n_{zout} . Any value of r_{max} and h_{max} is allowed for the convenience of the NITES operator and the calling application designer, provided $r_{max} \geq 5$ km, and $h_{max} \geq 100$ m. For example, the NITES operator may desire a coverage diagram which extends to a range of 500 kilometers (km). In addition to accommodating the desires of the operator, specification of such a convenient maximum range eases the burden for the application designer in determining incremental tick marks for the horizontal axis of the display.

Provided the value of the parameter *lerr12* is set to ‘.false.’, if the furthest environment profile range is less than r_{max} , the APM CSCI will automatically create an environment profile at r_{max} equal to the last profile specified, making the environment homogeneous from the range of the last profile specified to r_{max} . For example, a profile is input with an accompanying range of 450 km. If the NITES operator chooses an r_{max} of

500 km, the APM CSCI will continue loss calculations to 500 km, keeping the refractivity environment homogeneous from 450 km to 500 km.

If *lerr12* is set to ‘.true.’ and the furthest environment profile range is less than r_{max} , then an error will be returned in i_{error} from the APMINIT CSC. This is to allow the NITES CSCI application designer greater flexibility in how environment data is handled.

7.3 TERRAIN PROFILE DATA ELEMENT

The terrain profile should consist of linear piece-wise segments specified in terms of range/height pairs. All range values must be increasing, and the first terrain height value must be at range zero. General ground composition types can be specified (Table 4), along with corresponding ranges over which the ground type is to be applied. If ground type “User Defined” is specified ($igrnd_i = 7$), then numeric values of relative permittivity and conductivity must be given. If horizontal antenna polarization is specified, the APM CSCI will assume perfect conductivity for the entire terrain profile and will ignore any information regarding ground composition. If vertical antenna polarization is specified, then information regarding ground composition must also be specified.

The maximum height, h_{max} , must always be greater than the minimum height, h_{min} . Also, a value of h_{max} must be given such that it is larger than the maximum elevation height along a specified terrain profile.

Provided *lerr6* is set to ‘.false.’, if the furthest range point in the terrain profile is less than r_{max} , the APM CSCI will automatically create a height/range pair as part of the terrain profile at r_{max} with elevation height equal to the last height specified in the profile, making the terrain profile flat from the range of the last profile point specified to r_{max} . For example, a terrain profile is input where the last height/range pair is 50 meters (m) in height with an accompanying range of 95 km. If the NITES operator chooses an r_{max} of 100 km, the APM CSCI will continue loss calculations to 100 km, keeping the terrain profile flat from 95 km to 100 km with an elevation height of 50 m.

If *lerr6* is set to ‘.true.’ and the furthest range point is less than r_{max} , then an error will be returned in i_{error} from the APMINIT SU. This is to allow the NITES CSCI application designer greater flexibility in how terrain data is handled.

7.4 ACRONYM AND ABBREVIATIONS

The following table, Table 114, is a glossary of acronyms and abbreviations used within this document.

Table 114. Acronyms and abbreviations.

Term	Definition
MIN	Minimum of variables within parenthesis
MAX	Maximum of variables within parenthesis
AP	Angle trace function
APM	Advanced Propagation Model
Centibel	One-hundredth of the logarithm of a quantity
COMMON BLOCK	Allows two or more FORTRAN SUs to share variables without having to pass them as arguments
COS	Cosine function
CMPLX	Data conversion to complex number
CSCI	Computer software configuration item
dB	Decibel
decibel	10 times the logarithm of a quantity
EM	electromagnetic
FE	Flat earth
FFT	Fast-Fourier Transform
FORTRAN	Formula Translation
HP	Height trace function
IMAG	Imaginary part of complex number
INT	Integer value of
km	Kilometers
LOG ₁₀	Logarithm to base 10
LN	Natural logarithm
m	Meters
M	Modified refractivity units
MHz	MegaHertz
M-unit	Refractivity measurement unit
μV/m	Microvolts per meter
N/A	Not applicable
NINT	Round real number
PE	Parabolic Equation
p space	Phase space
RADA1	Angle trace function
radian	Unit of angular measurement
REAL	Real part of complex number
RO	Ray Optics

Table 114. Acronyms and abbreviations. (Continued)

Term	Definition
RP	Range trace function
SIGN	Sign transfer function
SIN	Sine function
SIN^{-1}	Inverse sine function
S/m	Conductivity unit Siemens per meter
$\text{Sin}(X)/X$	$\text{Sine}(X)/X$
SRS	Software Requirements Specification
SU	Software unit
TAN^{-1}	Inverse tangent function
NITES	Naval Integrated Tactical Environmental Subsystem
z-space	Height space

7.5 SDD VARIABLE NAME, FORTRAN VARIABLE NAME CROSS REFERENCE

The following table, Table 115, is a cross reference of variable names used within the body of this document and the FORTRAN variable names as used within the APM CSCI source code of Section 8, Appendix A. Included are the SDD variable name, its description, the FORTRAN source code name, and the designation of the FORTRAN COMMON BLOCK name, if applicable. Note – all dynamically allocated arrays are declared PUBLIC and are common to all SUs containing the APM_MOD module.

Table 115. Variable name cross reference.

SDD variable name	Description	FORTTRAN source code name	FORTTRAN common block name
a_0	Angle at start of ray trace step	a0	N/A
a_1	Angle at end of ray trace step	a1	N/A
a_2	Tangent angle for receiver height z_{outj}	ang2	N/A
a_{atz}	Local ray or propagation angle at height z_{lim} and range r_{atz}	aatz	APM_VAR
abs_{hum}	Absolute humidity near the surface	abshum	REFRACTIVITY
a_{crit}	Critical angle (angle above which no rays are trapped)	acrit	APM_VAR
a_{cut}	Tangent angle from antenna height to radio horizon	acut	APM_VAR
$adif$	Height differences between ant_{ref} and all output receiver heights	adif()	N/A
a_{ek}	Effective earth radius	aek	APM_VAR
a_{ekst}	4/3 effective earth's radius	aekst	N/A
$ainc_1$	Angular increment for ray tracing to determine grazing angles	ainc1	N/A
$ainc_2$	Angular increment for ray tracing to determine grazing angles	ainc2	N/A
$ainc_3$	Angular increment for ray tracing to determine grazing angles	ainc3	N/A
a_{launch}	Launch angle used which, when traced, separates PE and XO regions from the RO region	alaunch	APM_VAR
α	Source elevation angle	ang	N/A
α_d	Direct-path ray angle	alphad	APM_VAR
α_{dif}	The difference between current and previous outgoing propagation angles	angdif	N/A
α_{ld}	LOG of antenna pattern factor for α_d where α_d represents lowest direct ray angle in optical region	ald	N/A
α_{lim}	Elevation angle of the RO limiting ray	alflim	N/A
α_{pat}	Elevation angle relative to the antenna elevation angle	udif	N/A
α_r	Reflected-path ray angle	alphar	N/A
α_{ter}	Tangent angle from antenna height to current terrain height	alphax, terang	N/A
α_u	Maximum tangent ray angle from the source to the terrain peak along profile height	angu	N/A
$\alpha_{h,v}$	Surface impedance term for horizontal or vertical polarization	alphaq	APM_VAR

Table 115. Variable name cross reference. (Continued)

SDD variable name	Description	FORTRAN source code name	FORTRAN common block name
a_{mlim}	Elevation angle of RO limiting ray in radians. Used to initialize launch angle in the GETTHMAX SU	amlim	N/A
a_{mxcur}	Maximum local angle along the traced ray up to z_{lim} (with minimum limit a_{mlim})	amxcur	N/A
ant_{fac}	Antenna pattern parameter (depends on i_{pat} and μ_{bw})	afac	APM_VAR
ant_{ht}	Transmitting antenna height above local ground	antht	SYSTEMVAR
ant_{ko}	Height-gain value at source	antko	N/A
ant_{ref}	Transmitting antenna height relative to h_{minter}	antref	APM_VAR
a_s	Propagation angle for start of ray trace	as	N/A
$araft$	Array of angles after smoothing operation	araft()	N/A
$arbef$	Array of angles before smoothing operation	arbef()	N/A
a_{start}	Elevation angle at start of ray step	astart	N/A
a_{test}	Tangent angle used for automatic calculation of maximum propagation angle. Only used for modes $i_{hybrid} = 0, 2$.	atest	N/A
β	Terminal elevation angle	ab	N/A
β_d	Direct ray terminal elevation angle	betad	N/A
β_r	Reflected ray terminal elevation angle	betar	N/A
c_a	Wide-angle propagator correction term	cak	N/A
C_{1x}	Constant used to propagate c_{k1} by one range step in central difference algorithm	c1x	APM_VAR
C_{2x}	Constant used to propagate c_{k2} by one range step in central difference algorithm	c2x	APM_VAR
ck_1	Coefficient used in central difference form of DMFT	ck1	APM_VAR
ck_2	Coefficient used in central difference form of DMFT	ck2	APM_VAR
$cmft$	Coefficient used in backward difference form of DMFT	cmft	APM_VAR
$cmft_x$	Constant used to propagate $cmft$ by one range step in backward difference algorithm	cmft_x	APM_VAR
c_o	Speed of light (299.79×10^{-6} m/s)	c0	N/A
con	$10^{-6}k_o$	con	APM_VAR
cn_{p75}	Factor used in calculating $filtp$ array	cnp75	N/A
ct_1	Quantity defined in equ. 124 in EREPS 3.0 User's Manual NRaD TD 2648, pp. 106	ct1	N/A
ct_2	Quantity defined in equ. 125 in EREPS 3.0 User's Manual NRaD TD 2648, pp. 106	ct2	N/A

Table 115. Variable name cross reference. (Continued)

SDD variable name	Description	FORTTRAN source code name	FORTTRAN common block name
<i>curang</i>	Array of current local angles for each ray being traced in XO region	curang()	N/A
<i>curht</i>	Array of current local heights for each ray being traced in XO region	curht()	N/A
<i>curng</i>	Array of current local ranges for each ray being traced in XO region	curng()	N/A
ΔFd_{lo}^2	Difference in direct ray magnitude along Δx_{RO} below desired APM output point	dfsdllo	N/A
ΔFd_{hi}^2	Difference in direct ray magnitude along Δx_{RO} above desired APM output point	dfsldhi	N/A
ΔFr_{lo}^2	Difference in reflected ray magnitude along Δx_{RO} below desired APM output point	dfsrllo	N/A
ΔFr_{hi}^2	Difference in reflected ray magnitude along Δx_{RO} above desired APM output point	dfsrdhi	N/A
ΔH_o	Frequency gain function correction term defined in equ. 127 in EREPS 3.0 User's Manual NRaD TD 2648, pp. 106	delho	N/A
$\Delta \Omega_{hi}$	Difference in total phase lag angle along Δx_{RO} above desired APM output point	danghi	N/A
$\Delta \Omega_{lo}$	Difference in total phase lag angle along Δx_{RO} below desired APM output point	danglo	N/A
Δp	Mesh size in angle- (or p-) space	delp	APM_VAR
Δr_{grz}	PE range step used for calculation of grazing angles	drgrz	APM_VAR
Δr_{out}	Output range step	drout	APM_VAR
Δr_{PE}	PE range step	dr	APM_VAR
Δr_{PE2}	1/2 PE range step	dr2	APM_VAR
Δr_{temp}	Range step for ray tracing	drtemp	N/A
$\Delta \theta$	Angle difference between mesh points in p-space	dtheta	APM_VAR
Δx_{RO}	RO range interval	delxRO	APM_VAR
Δz_{out}	Output height increment	dzout	APM_VAR
Δz_{PE}	PE mesh height increment (bin width in z-space)	delz	APM_VAR
d_1	Range from source to tangent point	d1	N/A
d_2	Range from receiver to tangent point	d2	N/A
$d2s$	Array of tangent ranges for all output receiver heights over smooth surface	d2s()	N/A
$d\alpha$	$\frac{1}{2} \mu_{bwr}$	dalpha	N/A
<i>dielec</i>	2-dimensional array containing the relative permittivity and conductivity; <i>dielec</i> _{1,i} and <i>dielec</i> _{2,i} , respectively.	dielec(,)	N/A

Table 115. Variable name cross reference. (Continued)

SDD variable name	Description	FORTTRAN source code name	FORTTRAN common block name
$dx d\alpha$	Derivative of range with respect to elevation angle	dxda	N/A
$dx d\alpha_d$	Derivative of range with respect to α_d	dxdad	N/A
$dx d\alpha_r$	Derivative of range with respect to α_r	dxdar	N/A
$dz d\alpha_d$	Derivative of height with respect to α_d	dzdad	N/A
$dz d\alpha_r$	Derivative of height with respect to α_r	dzdar	N/A
e_k	Effective earth's radius factor	ek	APM_VAR
$envpr$	Complex [refractivity] phase term array	envpr()	N/A
ϵ_r	Relative permittivity	epsilon	N/A
η_s	Quantity defined in equ. 126 in EREPS 3.0 User's Manual NRaD TD 2648, pp. 106	etas	N/A
$f(\alpha)$	Antenna pattern factor for angle α	patfac	N/A
$f(\vartheta_1)$	Antenna pattern factor for angle ϑ_1	factr	N/A
$f(\alpha_d)$	Antenna pattern factor for direct ray	facd	N/A
$f(-\alpha_r)$	Antenna pattern factor for reflected ray	facr	N/A
$farray$	Field array to be propagated one range step in free space	farray()	N/A
F_d^2	Magnitude array, direct ray	dmagsq(.)	APM_VAR
F_{dB}	Propagation factor in dB	ff, facdb	N/A
$F_{dB st}$	Propagation factor in dB at previous range	pfdblst	N/A
$ffacz$	2-dimensional array containing propagation factor, range, and propagation angle at z_{lim}	ffacz(.)	N/A
$ffrout$	Array of propagation factors at each output range beyond r_{atz} and at height z_{lim}	ffrout()	N/A
$filt$	Cosine-tapered (Tukey) filter array	filt()	N/A
$filtp$	Array filter for spectral estimation calculations	filtp()	N/A
f_{MHz}	Frequency in MHz	freq	SYSTEMVAR
f_{rqg}	Frequency in MHz at which to perform grazing angle calculations	frqg	N/A
f_{norm}	Normalization factor	fnorm	APM_VAR
f_r	Fractional bin used for interpolation	fr	N/A
Fr^2	Magnitude array, reflected ray	rmagsq(.)	APM_VAR
$frac_{RO}$	RO range interval fraction (0.0 to 0.25)	fracRO	N/A
Fr_{atz}	Propagation factor in dB at range r_{atz} and height z_{lim}	pfratz	N/A
$frsp$	Complex free-space propagator term array	frsp()	N/A
fsl	Free space loss array for output ranges	fsl()	N/A

Table 115. Variable name cross reference. (Continued)

SDD variable name	Description	FORTTRAN source code name	FORTTRAN common block name
f_{sum}^2	Square of coherent sum of direct and reflected rays	ffac2	N/A
f_{ter}	Logical flag indicating if terrain profile has been specified: .true. = terrain profile specified .false. = terrain profile not specified	fter	APM_VAR
fv	Fraction range for profile interpolation	fv	N/A
γ_a	Surface specific attenuation	gammaa	REFRACTIVITY
γ_o	Oxygen absorption	gammao	N/A
γ_w	Water absorption	gammaw	N/A
$\Gamma_{h,v}$	Complex reflection coefficient for horizontal or vertical polarization	refcoef	N/A
gas_{att}	Gaseous absorption attenuation rate	gasatt	APM_VAR
gas_{loss}	Gaseous absorption loss at range r_{out}	gasloss	APM_VAR
gr	Intermediate M-unit gradient array, RO region	gr()	N/A
$grad$	2-dimensional array containing gradients of each profile used in XO calculations	grad(,)	N/A
g_{rd}	Refractivity gradient	grd	N/A
$grdum$	Array of refractivity gradients defined by profile $htdum$ and $refdum$	grdum()	N/A
h_0	Height at start of ray trace step	h0	N/A
h_1	Height at end of ray trace step	h1	N/A
H_1	Quantity defined in equ. 120 in EREPS 3.0 User's Manual NRaD TD 2648, pp. 106	hor1	N/A
H_2	Quantity defined in equ. 121 in EREPS 3.0 User's Manual NRaD TD 2648, pp. 106	hor2	N/A
$hfang$	Cut-back angles in degrees	hfang()	N/A
$hfangr$	Array of height-finder cut-back angles in radians	hfangr()	N/A
$hffac$	Cut-back antenna pattern factors	hffac()	N/A
h_{large}	Maximum height limit for last level in height/refractivity profiles	hlarge	N/A
$hlim$	Array containing height at each output range separating the RO region from the PE (at close ranges) and XO (at far ranges) regions	hlim()	N/A
h_{max}	Maximum output height with respect to mean sea level	hmax	INPUTVAR
h_{min}	Minimum output height with respect to mean sea level	hmin	INPUTVAR
h_{minter}	Minimum height of terrain profile	hminter	APM_VAR
h_{mref}	Height relative to h_{minter}	hmref	APM_VAR

Table 115. Variable name cross reference. (Continued)

SDD variable name	Description	FORTTRAN source code name	FORTTRAN common block name
<i>hmsl</i>	2-dimensional array containing heights with respect to mean sea level of each profile. Array format must be $hmsl_{i,j}$ = height of i^{th} level of j^{th} profile; $j=1$ for range-independent cases	<i>hmsl()</i>	N/A
h_o	Effective scattering height - defined in equ. 109 in EREPS 3.0 User's Manual NRaD TD 2648, pp. 105	<i>h0</i>	N/A
H_o	Frequency gain function defined in equ. 119 in EREPS 3.0 User's Manual NRaD TD 2648, pp. 106	<i>bigh</i>	N/A
<i>href</i>	Heights of refractivity profile with respect to y_{ref}	<i>href()</i>	N/A
h_s	Height for start of ray trace	<i>hs</i>	N/A
h_{start}	Starting height for ray trace to fill array <i>hlim</i>	<i>hstart</i>	N/A
<i>ht</i>	PE mesh height array of size n_{ft}	<i>ht()</i>	N/A
<i>htdum</i>	Height array for current interpolated profile	<i>htdum()</i>	N/A
<i>htemp</i>	Heights at which ray is traced to every range in <i>rtemp</i>	<i>htemp()</i>	APM_VAR
h_{ter}	Height of terrain at end of ray trace step	<i>hter</i>	N/A
h_{termax}	Maximum terrain height along profile path	<i>htermax</i>	N/A
h_{test}	Minimum height at which all trapping refractivity features are below	<i>htest</i>	N/A
<i>htfe</i>	Array containing the height at each output range separating the FE region from the RO region (full hybrid mode), or the FE region from the PE region (partial hybrid mode)	<i>htfe()</i>	N/A
h_{thick}	Thickness of highest trapping layer from all refractivity profiles	<i>hthick</i>	N/A
ht_{lim}	Maximum height relative to h_{minter}	<i>htlim</i>	APM_VAR
<i>htr</i>	2-dimensional array containing heights of each profile used in XO calculations	<i>htr()</i>	N/A
h_{trap}	Height of highest trapping layer from all refractivity profiles	<i>htrap</i>	N/A
ht_{ydif}	$ht_{lim} - y_{fref}$	<i>htydif</i>	APM_VAR
i_{alg}	Integer flag indicating which DMFT algorithm is being used: 0 = no DMFT algorithm will be used 1 = use central difference algorithm 2 = use backward difference algorithm	<i>ialg</i>	APM_VAR
i_{ap}	Index indicating when the local ray angle becomes positive in array <i>raya</i>	<i>iap</i>	APM_VAR
i_{err}	Return error code	<i>ierr</i>	N/A
i_{error}	Error flag – traps for various errors dependent on the calling SU	<i>ierror</i>	N/A

Table 115. Variable name cross reference. (Continued)

SDD variable name	Description	FORTTRAN source code name	FORTTRAN common block name
i_{extra}	Extrapolation flag for refractivity profiles entered in combination with terrain below below mean sea level 0 = extrapolate to minimum terrain height standard atmosphere gradient 1 = extrapolate to minimum terrain height using first gradient in profile	iextra	REFRACTIVITY
i_{flag}	Flag indicating whether to determine maximum FFT size n_{fft} based on given θ_{max} and z_{lim} or determine z_{lim} based on given θ_{max} and FFT size n_{fft} .	iflag	N/A
i_{flag}	Flag to indicate which transform to perform 0 = cosine transform 1 = sine transform -1 = deallocates all allocated arrays	iflag	N/A
i_{flag}	Integer flag indicating what region reflection coefficient is being computed 0 = FE and RO regions 1 = PE region	iflag	N/A
i_g	Counter indicating current ground type being modeled	ig	APM_VAR
i_{gPE}	Number of grazing angles computed from spectral estimation	igpe	APM_VAR
i_{gr}	Number of different ground types specified	igr	TERRAIN
i_{grz}	Number of grazing angles computed from ray trace	igrz	APM_VAR
i_{grad}	Index of current gradient level in <i>grad</i>	igrad	N/A
$igrd$	Integer indexes indicating at what refractive gradient level to begin ray tracing for next XO range step for each ray in XO region	igrd()	N/A
$igrnd$	Integer array containing ground type composition for given terrain profile - can vary with range. Different ground types are: 0 = sea water 1 = fresh water 2 = wet ground 3 = medium dry ground 4 = very dry ground 5 = ice at -1 degree C 6 = ice at -10 degree C 7 = user defined (in which case, values of relative permittivity and conductivity must be given).	igrnd()	N/A
i_{grz}	Number of grazing angles computed from ray trace	igrz	APM_VAR

Table 115. Variable name cross reference. (Continued)

SDD variable name	Description	FORTTRAN source code name	FORTTRAN common block name
i_{hmx}	Output range step index where height ht_{lim} is reached in array $hlim$	ihmx	APM_VAR
i_{hybrid}	Integer indicating which sub-models will be used: 0 = pure PE model 1 = full hybrid model (PE + FE + RO + XO) 2 = partial hybrid model (PE + XO)	ihybrid	APM_VAR
i_o	Starting index for $mpfl$ array: 0 = 1 st calculated output point is at surface 1 = 1 st calculated output point is at height Δz_{out}	io	APM_VAR
i_{org}	Integer flag indicating origin of calling SU 0 = called from APMINIT CSC 1 = called from TROPOINT SU	iorg	APM_VAR
i_{p1}	First output height point index in $zout$ where propagation loss will be computed at previous PE range	ip1	N/A
i_{p2}	First output height point index in $zout$ where propagation loss will be computed at current PE range	ip2	N/A
i_{pat}	Antenna pattern type 1 = Omni-directional 2 = Gaussian 3 = Sine(x)/x 4 = Cosecant-squared 5 = Generic height-finder 6 = User-defined height-finder 7 = User-defined antenna patter	ipat	SYSTEMVAR
i_{PE}	Number of PE range steps	ipe	APM_VAR
i_{peak}	Bin # in spectr corresponding to the peak magnitude	ipeak	N/A
i_{PEstp}	Counter indicating current PE range step	ipestp	N/A
i_{pl}	Polarization flag 0 = horizontal 1 = vertical	ipl	N/A
i_{pol}	Polarization flag: 0 = horizontal polarization 1 = vertical polarization	ipol	SYSTEMVAR
i_{quit}	Integer flag indicating to quit tracing current ray and begin again with a new launch angle	iquit	N/A
i_{ratz}	Index of output range step in which to begin storing propagation factor and outgoing angle for XO region	iratz	APM_VAR
i_{ROn}	Array index for next range in RO region	iROn	APM_VAR
i_{ROp}	Array index for previous range in RO region	iROp	APM_VAR

Table 115. Variable name cross reference. (Continued)

SDD variable name	Description	FORTTRAN source code name	FORTTRAN common block name
i_{rp}	Counter for current refractivity/gradient profile being used from <i>grad</i>	irp	N/A
i_{rps}	Starting index counter for refractivity profiles	irps	N/A
i_{rtemp}	Temporary number of range steps (used for ray tracing)	irtemp	APM_VAR
i_s	Counter for current profile	is	APM_VAR
i_{start}	Array index for height in RO region corresponding to ant_{ref}	istart	APM_VAR
i_{start1}	Refractivity level index within <i>htdum</i> at ant_{ref}	istart1	APM_VAR
i_{stp}	Current output range step index	istp	N/A
i_{sz}	Number of points over which to perform average smoothing	isz	N/A
i_{tp}	Number of height/range points in profile	itp	TERRAIN
i_{tpa}	Number of height/range points pairs in profile <i>tx</i> , <i>ty</i>	itpa	APM_VAR
i_{type}	Ray type (direct or reflected) flag 0 = direct 1 = reflected	itype	N/A
i_{xo}	Number of range steps in XO calculation region	ixo	APM_VAR
i_{xostp}	Current output range step index for XO calculations	ixostp	N/A
iz	Number of propagation factor, range, angle triplets stored in <i>ffacz</i>	iz	APM_VAR
iz_g	Number of output height points corresponding to local ground height at current output range rout	izg	APM_VAR
iz_{inc}	Integer increment for storing points at top of PE region (i.e., points are stored at every iz_{inc} range step)	izinc	APM_VAR
iz_{max}	Maximum number of points allocated for arrays associated with XO calculations	izmax	APM_VAR
j_{ae}	Ending index within <i>mpfl</i> of airborne loss values	jae	N/A
j_{as}	Starting index within <i>mpfl</i> of airborne loss values	jas	N/A
j_e	Ending receiver height index at which to compute troposcatter loss	je	N/A
j_{end}	Index at which valid loss values in <i>mpfl</i> end	jend	N/A
j_{fe}	Ending index within <i>mpfl</i> of FE loss values	jfe	N/A
j_{fs}	Starting index within <i>mpfl</i> of FE loss values	jfs	N/A
j_{max}	Array index for maximum output height in RO region	jmax	N/A
j_{min}	Array index for minimum output height in RO region	jmin	N/A
j_{pe}	Ending index within <i>mpfl</i> of PE loss values	jpe	N/A

Table 115. Variable name cross reference. (Continued)

SDD variable name	Description	FORTRAN source code name	FORTRAN common block name
j_{ps}	Starting index within <i>mpfl</i> of PE loss values	jps	N/A
j_{re}	Ending index within <i>mpfl</i> of RO loss values	jre	N/A
j_{rs}	Starting index within <i>mpfl</i> of RO loss values	jrs	N/A
j_s	Refractive profile index for start of ray trace	js	N/A
j_s	Starting receiver height index at which to compute troposcatter loss	js	N/A
j_{start}	Index at which valid loss values in <i>mpfl</i> start	jstart	N/A
$jt2$	Index counter for tx and ty arrays indicating location of receiver range	jt2	APM_VAR
j_{xe}	Index at which valid loss values in <i>mpfl</i> end	jxe	N/A
j_{xs}	Index at which valid loss values in <i>mpfl</i> start	jxs	N/A
j_{xstart}	Starting index within <i>mpfl</i> of XO loss values	jxstart	N/A
jz_{lim}	PE bin # corresponding to z_{lim} , i.e., $z_{lim} = jz_{lim} \Delta z_{PE}$	jzlim	APM_VAR
k_{abs}	Gaseous absorption calculation flag: 0 = no absorption loss 1 = compute absorption loss based on air temperature t_{air} and absolute humidity abs_{hum} 2 = compute absorption loss based on specified absorption attenuation rate γ_a	kabs	N/A
k_{bin}	Number of bins complex PE field is to be shifted	kbin	N/A
k_{hi}	k index above desired point	khi	N/A
k_{lo}	k index below desired point	klo	N/A
k_{max}	Array index for maximum angle in RO region at range x_{ROn}	kmax	APM_VAR
k_{minn}	Array index for minimum angle in RO region at range x_{ROn}	kminn	APM_VAR
k_{minp}	Array index for minimum angle in RO region at range x_{ROp}	kminp	APM_VAR
k_o	Free-space wave number	fko	APM_VAR
k_{temp}	Temporary k_{lo} value	klotmp	N/A
λ	Wavelength	wl	APM_VAR
L	Propagation loss	dloss	N/A
L_{dif}	Difference between propagation loss and troposcatter loss	dif	N/A
l_{duct}	Logical flag indicating if surface-based duct profile has been specified 'true' = surface-based duct exists 'false' = no surface-based duct exists	lduct	APM_VAR
$lerr6$	User-provided error flag that will trap on certain errors if set to 'true.'	lerr6	ERRORFLAG

Table 115. Variable name cross reference. (Continued)

SDD variable name	Description	FORTRAN source code name	FORTRAN common block name
<i>lerr12</i>	User-provided error flag that will trap on certain errors if set to '.true.'	lerr12	ERRORFLAG
<i>levap</i>	Logical flag indicating if evaporation duct profile has been specified 'true' = evaporation duct exists 'false' = no evaporation duct exists	levap	APM_VAR
<i>levels</i>	Number of levels in <i>gr</i> , <i>q</i> and <i>zrt</i> arrays	levels	APM_VAR
<i>L_{fs}</i>	Free space loss	fsloss	N/A
<i>lnew</i>	Temporary refractivity level counter	newl	N/A
<i>ln_{fft}</i>	Power of 2 transform size, i.e., $n_{fft}=2^{ln_{fft}}$	ln	APM_VAR
<i>ln_{min}</i>	Minimum power of 2 transform size	lnmin	APM_VAR
<i>ln_p</i>	Power of 2 transform size used in spectral estimation calculations; i.e., $n_p = 2^{ln_p}$	lnp	APM_VAR
<i>lvl</i>	Number of height levels in each profile used in XO calculations	lvl()	N/A
<i>lvlep</i>	Number of height/refractivity levels in profile <i>refdum</i> and <i>htdum</i>	lvlep	APM_VAR
<i>lvlp</i>	Number of height/refractivity levels in profiles	lvlp	REFRACTIVITY
<i>m</i>	Size of array <i>arbef</i>	m	N/A
<i>mpfl</i>	2-dimensional propagation factor and loss array	mpfl	N/A
μ_o	Antenna elevation angle in degrees	elev	SYSTEMVAR
μ_{or}	Antenna pattern elevation angle in radians	elv	APM_VAR
μ_{bw}	Antenna vertical beamwidth in degrees	bwidth	SYSTEMVAR
μ_{bwr}	Antenna vertical beamwidth in radians	bw	APM_VAR
μ_{lim}	Limiting elevation angle - no more than 10°	elv_lim	N/A
μ_{max}	Limiting angle for Sin(X)/X and generic height-finder antenna pattern factors	umax	APM_VAR
<i>n₃₄</i>	$\frac{3}{4} n_{fft}$	n34	APM_VAR
<i>n₄</i>	$\frac{1}{4} n_{fft}$	n4	APM_VAR
<i>nc²</i>	Array of complex dielectric constants	cn2()	N/A
<i>n_{fft}</i>	Transform size	n	APM_VAR
<i>n_{facs}</i>	Number of user-defined cut-back angles and cut-back pattern factors	nfacs	SYSTEMVAR
<i>nvl</i>	Number of levels in new profile	nvl	APM_VAR
<i>n_{m1}</i>	$n_{fft}-1$	nm1	APM_VAR
<i>no_{PE}</i>	Integer flag indicating if PE calculations are needed: 0 = PE calculations needed 1 = no PE calculations needed	nope	APM_VAR

Table 115. Variable name cross reference. (Continued)

SDD variable name	Description	FORTTRAN source code name	FORTTRAN common block name
n_{p34}	$\frac{3}{4} n_p$	np34	APM_VAR
n_{p4}	$\frac{1}{4} n_p$	np4	APM_VAR
n_p	Number of bins in upper PE region to consider for spectral estimation	npnts	APM_VAR
n_{prof}	Number of refractivity profiles	nprof	REFRACTIVITY
n_{ray}	Number of rays used for ray trace to determine grazing angles	nray	N/A
n_{rout}	Integer number of output range points desired	nrout	INPUTVAR
n_s	Transform size for spectral estimation calculations	ns	APM_VAR
n_{xo}	Number of rays traced, i.e., height points, in XO region	nxo	N/A
n_{zout}	Integer number of output height points desired	nzout	INPUTVAR
n_w	Number of wind speeds	nw	REFRACTIVITY
Ω	Total phase angle	phdif	N/A
Ω	Total phase angle array	omega(,)	N/A
ω_s	Interpolated wind speed	wnd	N/A
PE_{flag}	Flag to indicate use of PE algorithm only: 'true.' = only use PE sub-model 'false.' = use automatic hybrid model	peflag	INPUTVAR
p_{elev}	Sine of antenna elevation angle	pelev	APM_VAR
φ	Phase lag angle of reflected ray	rphase	N/A
pl_{cnst}	Constant used in determining propagation loss ($pl_{cnst} = 20 \log_{10}(2 k_o)$)	plcnst	APM_VAR
pl_d	Path length difference from range x for direct ray	pld	N/A
p_{elev}	Sine of antenna elevation angle	pelev	APM_VAR
pl_r	Path length difference from range x for reflected ray	plr	N/A
p_{mag}	Interpolated magnitude of complex PE field	pmag	N/A
$prfhxo$	2-dimensional array of propagation factor and heights for each ray traced in XO region to range r_{out}	prfh_xo	N/A
$profint$	Profile interpolated to every Δz_{PE} in height	profint()	N/A
ψ	Grazing angle	psi, angle	N/A
Ψ	Array of interpolated grazing angles at each PE range step	graze()	N/A
ψ_{lim}	Grazing angle of limiting ray	psilim	APM_VAR
ψ_{PE}	Array containing grazing angles computed from spectral estimation of PE field	grz_pe()	N/A

Table 115. Variable name cross reference. (Continued)

SDD variable name	Description	FORTTRAN source code name	FORTTRAN common block name
ψ_{ray}	2-dimensional array containing grazing angles and corresponding ranges computed from ray trace	grz_ray(.)	N/A
q	Intermediate M-unit difference array, RO region	q()	N/A
q_t	Quantity defined in equ. 128 in EREPS 3.0 User's Manual NRaD TD 2648, pp. 107	qt	N/A
r	Current PE range	r	N/A
r_0	Range at start of ray trace step	r0	N/A
r_1	Range at end of ray trace step	r1	N/A
r_1	Path length for direct-ray path	r1	N/A
r_1	Quantity defined in equ. 122 in EREPS 3.0 User's Manual NRaD TD 2648, pp. 106	r1	N/A
r_2	Path length for reflected-ray path	r2	N/A
r_2	Quantity defined in equ. 123 in EREPS 3.0 User's Manual NRaD TD 2648, pp. 106	r2	N/A
r_{range}	Range for profile interpolation	range	N/A
$ratio$	Fractional range term used for interpolation	ratiox	N/A
$ratio_k$	Fraction of one k index (0 to 1)	ratiok	N/A
r_{atz}	Range at which z_{lim} is reached (used for hybrid model)	ratz	APM_VAR
$raya$	Array containing all local angles of traced ray a_{launch} at each i_{temp} range	raya()	APM_VAR
r_{crit}	Minimum M-unit value above height ant_{ref}	rcrit	N/A
$rdif_1$	Range difference between adjacent terrain points	rdif1	N/A
$rdif_2$	Range difference between adjacent terrain points	rdif2	N/A
r_{difsum}	Sum of adjacent terrain point differences	rdifsum	N/A
rdt	Array of minimum ranges at which diffraction field solutions are applicable (for smooth surface) for all output receiver heights	rdt()	N/A
$refdum$	M-unit array for current interpolated profile	refdum()	N/A
$refmsl$	2-dimensional array containing refractivity with respect to mean sea level of each profile. Array format must be $refmsl_{i,j}$ = M-unit at i^{th} level of j^{th} profile; $j=1$ for range-independent cases	refmsl(,)	N/A
$refref$	Refractivity profile with respect to y_{ref}	refref()	N/A
r_f	Constant used for troposcatter calculations	rf	APM_VAR
$rfac1$	Propagation factor at valid output height points from PE field at range r_{last}	rfac1()	N/A
$refmsl$	2-dimensional array containing refractivity with respect to mean sea level of each profile. Array format must be $refmsl_{i,j}$ = M-unit at i^{th} level of j^{th} profile; $j=1$ for range-independent cases	refmsl(,)	N/A

Table 115. Variable name cross reference. (Continued)

SDD variable name	Description	FORTTRAN source code name	FORTTRAN common block name
<i>refref</i>	Refractivity profile with respect to y_{ref}	refref()	N/A
r_f	Constant used for troposcatter calculations	rf	APM_VAR
<i>rfac1</i>	Propagation factor at valid output height points from PE field at range r_{last}	rfac1()	N/A
<i>rfac2</i>	Propagation factor at valid output height points from PE field at range r	rfac2()	N/A
r_{fix}	Fixed range increment of terrain profile	rfix	N/A
r_{flat}	Maximum range at which the terrain profile remains flat from the source	rflat	N/A
r_{frac}	Ratio between adjacent terrain point differences	rfrac	N/A
<i>rgrnd</i>	Array containing ranges at which varying ground types apply	rgrnd()	N/A
r_{hor}	Radio horizon range	rhorr	APM_VAR
r_{hor1}	Minimum range at which diffraction field solutions are applicable - determined for 0 receiver height	rdhor1	N/A
R_k	Constant used to compute coefficients in central difference form of the DMFT	rk	APM_VAR
r_{last}	Previous PE range	rlast	N/A
r_{log}	$10 \log_{10}(\text{PE range } r)$	rlog	APM_VAR
<i>rlogo</i>	Array containing 20 times the logarithm of all output ranges	rlogo()	N/A
r_{loglst}	$10 \log_{10}(\text{previous PE range } r_{last})$	rloglst	APM_VAR
<i>rloss</i>	Propagation loss	rloss()	N/A
<i>rm</i>	Intermediate M-unit array, RO region	rm()	N/A
R_{mag}	Magnitude of reflection coefficient	rmag	N/A
r_{max}	Maximum specified range	rmax	INPUTVAR
r_{mid}	Range at which interpolation for range-dependent profiles is performed	rmid	N/A
rm_{max}	Maximum M-unit value of refractivity profile at range 0	rmmax	N/A
rm_{min}	Minimum M-unit value of refractivity profile at range 0	rmmin	N/A
rm_{tx}	M-unit value at height ant_{ref}	rmtx	APM_VAR
r_{mult}	PE range step multiplication factor	rmult	INPUTVAR
rn	Array of R_T to the i^{th} power (e.g., $rn_i = R_T^i$)	rn()	N/A
<i>rngout</i>	Array containing all desired output ranges	rngout()	N/A
<i>rngprof</i>	Ranges of each profile: $rngprof_i = \text{range of } i^{th} \text{ profile}$	rngprof()	N/A
<i>rngwind</i>	Ranges of wind speeds entered: $rngwind_i = \text{range of } i^{th} \text{ wind speed}$	rngwind	N/A

Table 115. Variable name cross reference. (Continued)

SDD variable name	Description	FORTRAN source code name	FORTRAN common block name
r_o	Current ending range for ray trace step	ro	N/A
r_{out}	Current output range	rout	N/A
r_{pest}	Range at which PE loss values will start being calculated	rpest	APM_VAR
r_s	Range for start of ray trace	rs	N/A
r_{skip}	Approximate range interval of skip zone if duct is present	rskip	N/A
r_{slope}	Ray slope used in determining reflection point over terrain	rslope	N/A
r_{sq}	Square of current output range	rsq	N/A
r_{sqk}	Earth curvature correction factor	rsqk	N/A
$rsqrd$	Array containing the square of all desired output ranges	rsqrd()	N/A
R_T	Complex root of quadratic equation for mixed transform method based on Kuttler's formulation	rt	APM_VAR
rt_1	rt multiplied by ant_{ref}	r1t	APM_VAR
$rtemp$	Range steps for tracing to determine maximum PE angle	rtemp()	APM_VAR
r_{tst}	Range at which to begin RO calculations (equal to 2.5 km)	rtst	N/A
ruf	Logical flag indicating if rough sea surface calculations are required 'true.' = perform rough sea surface calculations 'false.' = do not perform rough sea surface calculations	ruf	APM_VAR
ruf_{fac}	Factor used for wave height calculation	ruf_fac	APM_VAR
ruf_{ht}	Sea surface rms wave height	ruf_ht	APM_VAR
rv_1	Range of the previous refractivity profile	rv1	N/A
rv_2	Range of the next refractivity profile	rv2	APM_VAR
σ	Conductivity	sigma	N/A
s	Quantity defined equ. 110 in EREPS 3.0 User's Manual NRaD TD 2648, pp. 105	s	N/A
s_{bw}	Sine of antenna vertical beam width	sbw	APM_VAR
s_{gain}	Normalization factor used in starter field calculation	sgain	N/A
slp	Slope of each segment of terrain	slp()	N/A
sn_1	Term used in troposcatter loss calculation	sn1	N/A
$snref$	Surface refractivity	snref	N/A
$snref_{tx}$	Surface refractivity at transmitter	snref_tx	APM_VAR
$spectr$	Spectral amplitude of field	spectr()	N/A

Table 115. Variable name cross reference. (Continued)

SDD variable name	Description	FORTTRAN source code name	FORTTRAN common block name
ϑ_0	Array of angles used to determine common volume scattering angle	theta0()	N/A
ϑ_1	Tangent angle from source height	theta1	N/A
ϑ_2	Tangent angle from receiver height	theta2	N/A
ϑ_{1s}	Tangent angle from source (for smooth surface)	theta1s	APM_VAR
ϑ_{2s}	Array of tangent angles from all output receiver heights - used with smooth surface	theta2s()	N/A
ϑ_{1t}	Array of tangent angles from source height - used with terrain profile	th1()	N/A
θ_{max}	Maximum propagation angle in PE calculations	thetamax	N/A
ϑ_{mxg}	Maximum PE calculation angle for spectral estimation of grazing angles	thmxg	N/A
θ_{75}	75% of maximum propagation angle in PE calculations	theta75	APM_VAR
ϑ_{out}	Outgoing propagation angle determined at top of PE region	thout	N/A
θ	Common volume scattering angle	theta	N/A
θ_t	Angular interval limit for ray trace in determining grazing angles	degt	N/A
t_{air}	Air temperature near the surface	tair	REFRACTIVITY
$terx$	Range points of terrain profile	terx()	N/A
$tery$	Height points of terrain profile	tery()	N/A
th_{max}	Visible portion of maximum PE calculation angle	thmax	INPUTVAR
t_{loss}	Troposcatter loss in dB	tloss	N/A
$tlst$	Troposcatter loss term	tlst	N/A
$tlst_s$	Troposcatter loss term for smooth surface case	tlsts	APM_VAR
$tlst_{wr}$	Troposcatter loss term used in TROPOSCAT SU	tsltwr	APM_VAR
T_{ropo}	Troposcatter calculation flag: '.false.' = no troposcatter calcs '.true.' = troposcatter calcs	tropo	INPUTVAR
$twoka$	Twice the effective earth's radius	twoka	APM_VAR
$twoka_{down}$	Twice the effective earth radius for downward path	twoka_down	APM_VAR
tx	Range points of terrain profile	tx()	N/A
ty	Adjusted height points of terrain profile	ty()	N/A
tyh	Adjusted height points of sampled terrain profile at every PE range step	tyh()	N/A
U	Complex field at current PE range r	u()	N/A

Table 115. Variable name cross reference. (Continued)

SDD variable name	Description	FORTTRAN source code name	FORTTRAN common block name
<i>Udum</i>	Dummy array used for temporary storage of real or imaginary part of complex PE field array <i>U</i>	udum()	N/A
<i>Ulast</i>	Complex field at previous PE range r_{last}	ulst()	N/A
<i>w</i>	Difference equation of complex PE field	w()	N/A
<i>wind</i>	Array of wind speeds	wind()	N/A
<i>x</i>	Current output range	x	N/A
<i>x</i>	Field array to be transformed - dimensioned $2^{n_{fft}}$ in calling SU	x()	N/A
<i>xdum</i>	Real part of complex field array	xdum()	N/A
<i>xO_{con}</i>	Constant used in determining v_{out}	xocon	APM_VAR
<i>xp</i>	Real part of spectral field	xp()	N/A
<i>x_r</i>	Terminal range - called x_{ROn} in RO CALC SU	rout	N/A
<i>x_{reflect}</i>	Range at which ray is reflected	xreflect	N/A
<i>xROn</i>	Next range in RO region	xROn	APM_VAR
<i>xROp</i>	Previous range in RO region	xROp	APM_VAR
<i>x_{temp}</i>	Temporary range in ray trace step	xtemp	N/A
<i>x_{sum}</i>	Running sum of range during ray trace	xsum	N/A
<i>xx</i>	Fractional range for interpolation	xx	N/A
<i>y_{ch}</i>	Height of terrain at the current PE range relative to hm_{ref}	ych	N/A
<i>y_{cur}</i>	Height of ground at current range <i>r</i>	ycur	APM_VAR
<i>y_{curm}</i>	Height of ground midway between last and current PE range	ycurm	APM_VAR
<i>y_{diff}</i>	$y_{cur} - y_{last}$	ydiff	N/A
<i>ydum</i>	Imaginary part of complex field array	ydum()	N/A
<i>y_{fref}</i>	Ground elevation height at source	yfref	APM_VAR
<i>y_{last}</i>	Height of ground at previous range r_{last}	ylast	APM_VAR
<i>y_{lh}</i>	Height of terrain at the previous PE range relative to hm_{ref}	ylh	N/A
<i>ym</i>	Particular solution of difference equation	ym()	N/A
<i>yp</i>	Imaginary part of spectral field	yp()	N/A
<i>y_{ref}</i>	Ground elevation height at current range	yref	N/A
<i>z_d</i>	Terminal height of direct ray	zd	N/A
<i>z_{int}</i>	Interpolated terrain elevation at current output range	zint	N/A
<i>z_k</i>	Height of k^{th} RO index	zk	N/A
<i>z_{lim}</i>	Height limit for PE calculation region	zlim	APM_VAR
<i>z_{limt}</i>	$ht_{lim} \cdot 10^{-5}$	zlimt	N/A
<i>z_{max}</i>	Total height of the FFT/PE calculation domain	zmax	APM_VAR

Table 115. Variable name cross reference. (Continued)

SDD variable name	Description	FORTRAN source code name	FORTRAN common block name
<i>zout</i>	Array containing all desired output heights referenced to h_{minter}	zout()	N/A
<i>zoutma</i>	Array output heights relative to “real” ant_{ref}	zoutma()	N/A
<i>zoutpa</i>	Array output heights relative to “image” ant_{ref}	zoutpa()	N/A
<i>z_r</i>	Receiver height	height, zr	N/A
<i>zro</i>	Array of output heights in RO region	zro()	N/A
<i>zrt</i>	Intermediate height array, RO region	zrt()	N/A
<i>z_{test}</i>	Height in PE region that must be reached for hybrid model	ztest	N/A
<i>z_{tol}</i>	Height tolerance for Newton's method	ztol	APM_VAR

APPENDIX A

FORTRAN SOURCE CODE FOR APM CSCI

A.1 SUBROUTINE APMINIT

```

! Version 1.3.1
! Author: Amalia E. Barrios
!         SPAWARSYSCEN SAN DIEGO D858
!         49170 Propagation Path
!         San Diego, CA 92152-7385
!         e-mail: barrios@spawar.navy.mil
!         phone: (619) 553-1429
!         fax: (619) 553-1417

! Summary: These routines model tropospheric radiowave propagation over
!          variable terrain and calculates propagation loss vs. height and
!          range. Propagation loss is displayed in dB contours on a height vs.
!          range plot. APM is based on the Radio Physical Optics (RPO) model
!          developed by Herb Hitney (SPAWARSYSCEN SAN DIEGO) and the Terrain
!          Parabolic Equation Model (TPEM) developed by Amalia Barrios
!          (SPAWARSYSCEN SAN DIEGO). The parabolic equation sub-model is based
!          on the split-step Fourier PE method and was originally developed
!          from an early PE model called PEPC, written by Fred Tappert.

!          Propagation loss over variable terrain is modeled by shifting
!          the field an appropriate number of bin widths correspond-
!          ing to the height of the ground. The field is determined using the
!          smooth earth PE method. A hybrid capability is also included for
!          limited cases (low antenna heights and/or initial flat terrain).
!          The hybrid model consists of a flat earth (FE) region at very high
!          angles, a ray-optics (R0) model at intermediate angles, and
!          the split-step PE model below the lowest R0 angle. An extended
!          optics model (X0) is used at heights above the PE region
!          and at ranges beyond the R0 region.

! *****

! Variables in small letters in parameter lists are variables that are input
! or passed to called subroutines. Variables in CAPS in parameter lists are
! returned from the called subroutines.

! ***** SUBROUTINE APMINIT *****

! Module Name: APMINIT

! Module Security Classification: UNCLASSIFIED

! Purpose:  Initializes all variables used in APM subroutines for FE, R0,
!            and PE calculations. After initial units conversions have been
!            done, all calculations are in metric units. Height and range
!            values are in meters and angles are in radians.

! Version Number: 1.3.1

! INPUTS:
!   Argument List: NONE
!   Common:  ABSHUM, ANTHT, BWIDTH, ELEV, FREQ, GAMMAA, HMAX, HMIN, IGR, IPAT,
!            IPOL, ITP, NW, LERR6, LERR12, LVLP, NFACS, NPROF, NROUT, NW,
!            NZOUT, PEFLAG, RMAX, TAIR, THMAX, TROPO
!   Public:  DIELEC(), HFANG(), HFFAC(), HMSL(), IGRND(), REFMSL(), RGRND(),
!            RNGPROF(), RNGWIND(), TERX(), TERY(), WIND()
!   Parameters: PI
!   Data:  C0, DEG5, ISM, RADC, RTST

! OUTPUTS:
!   Argument List: IXOSTP, IERROR
!   Common:  Most variables in common block APM_VAR
!   Public:  All public arrays.

```

```

! Modules Used: APM_MOD

! Calling routines: MAIN DRIVER PROGRAM

! Routines called:
!   APM Specific: ALLARRAY_APM, ALLARRAY_PE, ALLARRAY_RO, ALLARRAY_XORUF,
!                 ALN_INIT, DIEINIT, FFTPAR, FILLHT, GASABS, GET_K, GETGRAZE,
!                 GETMODE, GETTHMAX, GRAZE_INT, INTPROF, PEINIT, PROFREF,
!                 REFINIT, REMDUP, TERINIT, TROPOINT
!   Intrinsic: ALLOCATE, ALLOCATED, ANY, DCOS, DABS, DATAN, DBLE, DEALLOCATE,
!              DLOG10, DMAX1, DMIN1, DSIGN, DSIN, DSQRT, IDINT, IDNINT, MAX0, MIN0

! GLOSSARY: See universal glossary for common variables, data variables
!           public arrays, and parameters.

!   Input Variables: NONE

!   Output Variables:
!     IXOSTP = Index of output range step at which XO model is to be applied.
!     IERROR = Integer value that is returned if any errors exist in input
!             data:
!       -6 : Last range in terrain profile is less than RMAX.
!             (will only return this error if error flag LERR6
!             is set to .TRUE.).
!       -7 : Specified cut-back angles are not increasing. This is
!             only tested for user-defined height-finder antenna
!             pattern.
!       -8 : HMAX is less than maximum height of terrain profile.
!       -9 : Antenna height w.r.t. msl must be less than maximum
!             height HMAX.
!       -10 : Beamwidth is less than or equal to zero for directional
!             antenna pattern.
!       -11 : Number of antenna pattern or power reduction factors and
!             angles (IPAT=6 or 7) is less than or equal to 1. For
!             IPAT=6, NFACS must be at least 1; for IPAT=7, NFACS must
!             be at least 2.
!       -12 : Range of last refractivity profile entered (for range
!             dependent case) is less than RMAX. (This is returned
!             from subroutine REFINIT). Will only return this error
!             if error flag LERR12 is set to .TRUE.).
!       -13 : Height of first level in any user-specified refrac-
!             tivity profile is greater than 0. First height must
!             be at m.s.l. (0.) or <0. if below m.s.l.
!       -14 : Last gradient in any refractivity profile entered is
!             negative. (This is returned from REFINIT).
!       -17 : Range points of terrain profile are not increasing.
!       -18 : First range point is not 0.
!       -19 : Elevation angle specified (for IPAT > 1) must be less than
!             10 degrees.
!       -25 : Specified PE-only flag but did not specify a maximum
!             PE calculation angle.
!             -41 : Transmitter height is less than 1.5 m.
!       -42 : Minimum height input by user (HMIN) is greater then
!             maximum height (HMAX).
!       -43 : Transform size is greater than 2**30. Lower antenna height
!             or frequency. Check geometry between antenna height and
!             any terrain features.
!             -44 : Combination of frequency and antenna beamwidth results
!                   in antenna below the surface. Increase frequency
or
!                   beamwidth for valid combination. CAN OVERRIDE WITH
PE-ONLY
!                   OPTION BUT MUST SPECIFY MAXIMUM PE PROPAGATION
ANGLE.
!                   -45 : Wind speed specified is greater than 10 m/s. CAN OVERRIDE
!                   WITH PE-ONLY OPTION BUT MUST SPECIFY MAXIMUM PE
PROPAGATION
!                   ANGLE.

!   Local Variables:

```

```

!      ALFLIM = Elevation angle of RO limiting ray in radians. Used to
!              initialize launch angle in GETTHMAX routine.
!      ANGU = Maximum tangent ray angle from source to terrain peak
!            along profile path.
!      ATEST = Tangent angle used for automatic calculation of maximum
!            propagation angle. Only used for modes IHYBRID = 0, 2.
!      DIAM = Physical antenna diameter required for specified vertical beamwidth
!            and frequency - used only for directional antenna patterns.
!      FRQG = Frequency in MHz at which to perform grazing angle calcs.
!      HMX = Maximum height to use when computing a maximum propagation
!            angle for PE calculations. Only used for modes IHYBRID=0,2.
!      HTERMAX = Maximum terrain height along profile path in meters.
!      HTEST = Minimum height in meters at which all trapping
!            refractivity features are below (includes some slop).
!      HTHICK = Thickness in meters of highest trapping layer from all
!            refractivity profiles.
!      HTRAP = Height of highest trapping layer in meters from all
!            refractivity profiles.
!      KABS = Integer flag indicating whether or not to compute gaseous
!            absorption loss. KABS=0 no absorption loss; KABS=1 compute
!            absorption loss based on air temperature TAIR and absolute
!            humidity ABSHUM; KABS=2 compute absorption loss based on specified
!            absorption attenuation rate GAMMAA.
!      RFIX = If terrain profile points are equally spaced, this is
!            automatically determined and range spacing is set to RFIX,
!            otherwise, RFIX = 0.
!      RFLAT = Maximum range in meters at which the terrain profile
!            remains flat from the source.
!      RMMAX = Maximum M-unit value (x10e-6) of refractivity profile at
!            range 0.
!      RMMIN = Minimum M-unit value (x10e-6) of refractivity profile at
!            range 0.
!      THETAMAX = Maximum propagation angle used in PE calcs.
!      THMXG = Maximum PE propagation angle for grazing angle calcs (3 degrees
!            + 1/3 for filtering).
!      ZTEST = This is the minimum height at which the PE model must
!            reach in order to contain all necessary refractivity and/or
!            terrain features.

```

```

subroutine apminit( IXOSTP, IERROR )

```

```

use apm_mod

```

```

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

```

```

data sdeg10 / .173648177d0 /      !Sine of 10 degrees
data sdeg15 / .258819045d0 /      !Sine of 15 degrees
data frqg, thmxg / 10000.d0, 6.981317d-2 /      !10 GHz, 3/.75 degrees

```

```

! Initialize variables.

```

```

pi2 = .5d0 * pi
ierror = 0
nope = 0
ism2 = ism / 2
ek = 0.
thetamax = 0.
kabs = 0
gasatt = 0.d0
rpest = 0.d0
ipe = 0
terang = -99.d0
iz = 1
ruf_fac = 0.
igrz = 0
ruf = .false.
ihybrid = -1
ixo = 0
ixostp = 0
io = 1 - ipol

```

```

! Check to see that actual non-zero wind speed values have been provided. If
! not, then set roughness flag to '.false.' Also check for wind speed limit of
! 10 m/s.

if( nw .gt. 0 ) ruf = any( wind .gt. 1.d-2 )
if( ruf ) then
    if( (any( wind .gt. 10.d0 )) .and. (.not. peflag)) ierror = -45
end if

! Initialize flags for absorption calculations.

if(( tair .ne. 0. ) .or. ( abshum .ne. 0. )) kabs = 1
if( gammaa .ne. 0. ) kabs = 2

if( hmin .ge. hmax ) then
    ierror = -42
    return
end if

!Perform error and limit checking only if PEFLAG has not been set (i.e., if
!using hybrid methods).

if( .not. peflag ) then

! Check on value of transmitter antenna height.

    if( antht .lt. 1.5d0 ) ierror = -41
    if( ierror .ne. 0 ) return

end if

! Put lower limit on HMAX and RMAX

rmax = dmax1( rmax, 5000.d0 ) !Set max. range to no less than 5 km.
hmax = dmax1( hmax, 100.d0 ) !Set max. height to no less than 100 m.
hmin = dmin1( hmin, hmax-100.d0 )

dzout = (hmax-hmin) / dble( nzout )
drout = rmax / dble( nrout )

itpa = itp + 1

call allarray_apm( IERROR )
if( ierror .ne. 0 ) return

! Calculate constants used to determine antenna pattern factor
! IPAT = 1 -> omni
! IPAT = 2 -> gaussian
! IPAT = 3 -> sinc x
! IPAT = 4 -> csc**2 x
! IPAT = 5 -> generic height-finder
! IPAT = 6 -> user-defined height-finder
! IPAT = 7 -> user-defined antenna pattern

if(( ipat .eq. 7 ) .and. ( nfacs .le. 1 )) ierror = -11
if(( ipat .eq. 6 ) .and. ( nfacs .eq. 0 )) ierror = -11
if( ierror .ne. 0 ) return

if(( ipat .eq. 6 ) .or. ( ipat .eq. 7 )) then
    hfangr = hfang * radc
    do i = 1, nfacs-1
        if( hfangr(i+1) .lt. hfangr(i) ) ierror = -7
    end do
    if( ierror .ne. 0 ) return
end if

! Check for limits on beamwidth and elevation angle.

if((ipat .gt. 1) .and. (ipat .ne. 7) .and. (.not. peflag)) then
    if( bwidth .le. 1.d-4 ) ierror = -10

```

```

    if( dabs(elev) .gt. 10.d0 ) ierror = -19
end if
if( ierror .ne. 0 ) return

if(( ipat .eq. 1 ) .or. ( ipat .eq. 7 )) bwidth = 45.    !For R0 calculations.

bw = bwidth * radc
elv = elev * radc
bw2 = .5d0 * bw
if( ipat .eq. 2 ) then    !Gaussian
    afac = .34657359d0 / (dsin( bw2 ))**2
    pelev = dsin( elv )
elseif( ipat .eq. 4 ) then    !CSC**2
    sbw = dsin( bw )
elseif(( ipat .ne. 1 ) .and. ( ipat .ne. 7 )) then
    afac = 1.39157 / dsin( bw2 )
    a = pi / afac
    umax = datan( a / dsqrt(1.d0 - a*a) )
    umax3 = 3. * umax
end if

! Test for validity of combination of antenna parameters. If running in
! hybrid mode, then frequency and beamwidth combination must not produce
! an antenna radius greater than the antenna height.

if(( .not. peflag ) .and. ( ipat .ne. 1 ) .and. ( ipat .ne. 7 )) then
    diam = 1.2d0 * c0 / (freq * bw)
    if( .5*diam .gt. antht ) ierror = -44
end if
if( ierror .lt. 0 ) return

! Initialize terrain information.

call terinit( ANGU, RFIX, RFLAT, HTERMAX, IERROR )
if( ierror .ne. 0 ) return

!Initialize mixed transform algorithm flag

ialg = 0
if(( io .eq. 0 ) .or. ( ruf )) then
    ialg = 2                                !backward difference
    if( freq .lt. 400.d0 ) ialg = 1    !central difference
end if

! Initialize output range arrays.

rngout = ((i, i=1, nrout)/) * drout
rlogo = 20. * dlog10( rngout )

! Initialize variables for all hybrid modes.

lnmin = 10
if( .not. peflag ) then

    rsqrd = rngout * rngout                !Used in AIRBORNE and FEM routines.

! Determine what hybrid model(s) to use.

    ihybrid = 0                            ! Use airborne hybrid (FE+PE) model
    if( antht .le. 100.d0 ) then
        ihybrid = 1                        ! Use full hybrid mode

! If the first 2.5 km of terrain profile is not flat then use partial
! hybrid mode (PE + X0).

        if(( fter ) .and. ( rflat .le. rtst+1.d-3 )) ihybrid = 2
    end if

! Allocate arrays, if necessary, for R0 calcs.

    if( ihybrid .eq. 1 ) call allarray_ro( IERROR )

```

```

        if( ierror .ne. 0 ) return
    end if

    ! Setup output height arrays with respect to HMINTER.

    yfref = 0.
    if( fter ) yfref = ty(1)

    zout = ((i, i=0, nzout)/) * dzout + hmref
    if( ihybrid .eq. 1 ) zro = zout - yfref

    ! Initialize refractivity arrays.

    call reinit( HTRAP, HTHICK, RMMIN, RMMAX, IERROR )
    if( ierror .ne. 0 ) return

    if( .not. peflag ) then

        !Used in AIRBORNE and FEM routines

        zoutma = zout - antref
        zoutpa = zout - yfref + antht

        ! Compute grazing angle limit based on 2.5 times Reed & Russell
        ! (p. 140) limit, but not less than .002 rad. Double this value if
        ! more than one profile was entered, then adjust for trapping
        ! effects. Compute corresponding R0 elevation angle limit at
        ! transmitter, ALFLIM.

        psilim = dmax1( .002d0, .04443d0 / (freq ** .3333333) )
        IF (nprof .GT. 1) psilim = 2. * psilim
        psilim = psilim + dSQRT(dABS(2.d0 * (rmmax - rmmmin)))
        alflim = dSQRT(dABS(psilim ** 2 + 2. * (rmtx - refdum(0)*1.d-6)))
        alflim = dmax1( alflim, acrit )

        ! Define height tolerance for Newton's method.

        ztol = .05

        ! Initialize range and index variables for R0 region.

        xR0n = 0.
        iR0p = -1

        ! Determine the minimum height the PE model must reach.

        htest = htrap + hthick

        if( ihybrid .eq. 1 ) then
            ztest = dmax1( htest, 1.2d0*htermmax )
        else
            ztest = dmax1( htlim, antref )
            hmx = ztest + antref + rmax**2 / (2. * aekst)
            atest = datan( hmx / rmax )
            alflim = dmax1( alflim, atest )
        end if

        !Determine effective earth radius term.

        call get_k( 0 )
    else

        ztest = htlim

    end if

    ! If wind speed has been specified (NW>=1), then perform entire PE run for smooth
    ! surface (0 wind speed), horizontal polarization, fixed frequency. Determine
    ! grazing angles at each PE range and store in GRAZE().

```



```

if( ruf ) then

    wl = c0 / frqg
    Fko = 2. * pi / WL
    fko2 = 2. * fko
    thmxg = dmax1( thmxg, angu )

    call fftpar( lnmin, wl, thmxg, 0, ZDUM, ZMAX, DELZ, LN, N, IERROR )
    if( ierror .ne. 0 ) return

    drgrz = 0.d0
    call peinit( rfix, 0, 0, IERROR )
    if( ierror .ne. 0 ) return
    drgrz = dr          !Store PE range step used for grazing angle calcs.

    npnts= 16
    lnp = 9
    ns = 2**lnp
    nsm1 = ns - 1
    np4 = npnts/4
    np34 = 3 * np4
    cnp75 = pi / dble(np4)

    call allarray_xoruf( IERROR )
    if( ierror .ne. 0 ) return

    filtp = .5 + .5 * dcos( ((i, i=0,np4)/) * cnp75 )
    xocon = wl / ns / 2. / delz

! Determine grazing angles.

    call getgraze( htrap, thmxg, IERROR )
    if( ierror .ne. 0 ) return

    if((fter) .or. ( nprof .gt. 1 )) then

! Re-initialize all variables associated with refractivity arrays used in
! PE calcs.

        is = 1
        rv2=rngprof(is)
        refdum(0:lvlp) = refmsl(0:lvlp, is)
        htdum(0:lvlp) = hmsl(0:lvlp, is)
        lvlp = lvlp

! Remove any duplicate levels in first profile and adjust HTDUM() and
! REFDUM() to minimum terrain height.

        call remdup
        call profref( hminter, 0 )
    end if

else

    if( allocated( graze ) ) deallocate( graze, stat=ierror )
    allocate( graze(0:1), stat=ierror )
    if( ierror .ne. 0 ) return
    graze = 0.
    igrz = 1

end if

! Compute wavelength and wavenumber for "real" run.

wl = c0 / freq
Fko = 2. * pi / WL
fko2 = 2. * fko

! Initialize dielectric ground constants.

```

```

ig = 1
call dieinit

! Initialize wave height variables.

if( ruf ) then
  ruf_fac = 6.408849d-2 / wl          !4 * pi * .0051 conversion
  ruf_ht = ruf_fac * wind(1)**2
end if

if( peflag ) then

  if( thmax .le. 1.d-3 ) ierror = -25
  if( ierror .ne. 0 ) return

  theta75 = thmax * radc
  thetamax = theta75 / .75
  ilfac = idint( theta75 / deg5 )
  lnmin = lnmin + ilfac

  call fftpar( lnmin, wl, thetamax, 0, ZTEST, ZMAX, DELZ, LN, N, IERROR )
  zlim = dmin1( htlim, ztest ) !in case ZLIM > calculation height domain

!Initialize PE variables for "real" run.

  call peinit( rfix, ipol, 1, IERROR )
  if( ierror .ne. 0 ) return

! Initialize mixed transform constants for start of PE calculations

  if( ialg .gt. 0 ) call aln_init( pi2, 0.d0 )
  rpest = 0.d0

else

  alim_v = alimv * radc  !*** FOR SPAWAR USE ONLY *****

  if( ihybrid .eq. 0 ) then

! For airborne hybrid model, maximum PE angle is determined from angles computed in
! GET_K routine.

    theta75 = dmax1( alim_v, theta75 )
    thetamax = theta75 / .75

    ilfac = idint( theta75 / deg5 )
    lnmin = lnmin + ilfac

    call fftpar( lnmin, wl, thetamax, 0, ZTEST, ZMAX, DELZ, LN, N, IERROR )
    if( ierror .ne. 0 ) return
    zlim = ztest

  else

! Determine the maximum PE propagation angle needed to get to AT LEAST this height.
! Also initialize all associated PE variables.

    call getthmax( htest, htermax, rflat, ztest, alflim, alim_v, THETAMAX, IERROR )
    if( ierror .ne. 0 ) return

  end if

  zlim = dmin1( htlim, zlim ) !in case ZLIM > calculation height domain

! For calculation modes IHYBRID = 1 or 2, initialize all variables for use
! in XO calculations.

  if( ihybrid .ne. 0 ) then

    if ( zlim .le. (htlim - zlim*1.d-5) ) then

```

```
! Get bin # within calculation domain (with respect to HMINTER) at
! which to perform spectral estimation. From JZLIM to JZLIM-NPNTS.
```

```
  jzlim = idint( zlim / delz )
  zlim = dble( jzlim ) * delz
```

```
!Determine RATZ and IRATZ.
```

```
  j = iap
  id = 1
  do while( j .le. irtemp )
    if( htemp(j) .gt. zlim ) exit
    if( htemp(j) .gt. htdum(id) ) id = id + 1
    j = j + 1
  end do
  ira = max0( 1, j-1 )
  idg = id - 1
  grd = grdum(idg)

  rad = raya(ira)**2 + 2. * grd * ( zlim - htemp(ira) )
  aatz = 0.
  if( rad .gt. 0. ) aatz = dsign( 1.d0, raya(ira) ) * dsqrt( rad )
  ratz = rtemp(ira) + (aatz - raya(ira)) / grd
  if(( ratz .lt. rmax ) .and. ( zlim .lt. htlim )) then
    k = 1
    do while( rngout(k) .lt. ratz )
      k = k + 1
    end do
    iratz = min0( nrout, k )
    ixostp = iratz
  end if
else
  iratz = nrout + 1
  ratz = 2. * rmax
end if

end if
ixo = ixostp

if( nope .eq. 0 ) then
```

```
!Initialize PE variables for "real" run.
```

```
  call peinit( rfix, ipol, 1, IERROR )
  if( ierror .ne. 0 ) return
```

```
! Determine number of points that will be stored in FFACZ(,), and
! allocate and initialize all arrays associated with extended optics calcs.
```

```
  if( ixo .gt. 0 ) then
    rmxdif = rmax - ratz
    niz = idnint( rmxdif / dr )
    izmax = niz / izinc + 4           !add some slop
```

```
! Initialize variables and filter array for spectral estimation.
```

```
  npnts = 8
  if(( fter ) .or. ( ruf )) npnts= 16
  lnp = 8
  ns = 2**lnp
  nsm1 = ns - 1
  np4 = npnts/4
  np34 = 3 * np4
  cnp75 = pi / dble(np4)

  call allarray_xoruf( IERROR )
  if( ierror .ne. 0 ) return

  filtp = .5 + .5 * dcos( /(i, i=0,np4)/) * cnp75 )
  xocon = wl / ns / 2. / delz
end if
```

```

! Initialize mixed transform constants for start of PE calculations
      if( ialg .gt. 0 ) call aln_init( pi2, 0.d0 )
    end if

! Now fill height array separating flat earth region from R0 region.
! The height is determined and stored at each output range step.

      call fillht

end if

ylast = 0.
if(( fter ) .and. ( nope .eq. 0 )) ylast = tyh(0)
ycurm = 0.
ycur = 0.

!Determine/interpolate grazing angles such that all grazing angles coincide with
!"real" PE range step.

if( ruf ) call graze_int( rflat, IERROR )
if( ierror .ne. 0 ) return

! Initialize troposcatter variables.

if( tropo ) call tropoint

!Loss term - add to 20log(r) to get free space loss.

plcnst=20.*dlog10(fko2)

!Initialize free-space loss array.

fsl = rlogo + plcnst

! Determine attenuation rate due to gaseous absorption in dB/m.

if( kabs .eq. 1 ) call gasabs
if( kabs .eq. 2 ) gasatt = gammaa * 1.d-3 !Conversion for range in m.

!Deallocate arrays no longer needed.

if( allocated( tx ) ) deallocate( tx, stat=kerror )
if( kerror .ne. 0 ) ierror = kerror

if( allocated( ty ) ) deallocate( ty, stat=kerror )
if( kerror .ne. 0 ) ierror = kerror

if( allocated( slp ) ) deallocate( slp, stat=kerror )
if( kerror .ne. 0 ) ierror = kerror

end subroutine apminit

```

A.1.1 Subroutine ALLARRAY_APM

```

!***** SUBROUTINE ALLARRAY_APM *****

! Module Name: ALLARRAY_APM

! Module Security Classification: UNCLASSIFIED

! Purpose: This routine allocates and initializes all dynamically dimensioned
!          arrays associated with APM general info, terrain, refractivity,
!          and troposcatter arrays.

! Version Number: 1.3.0

! INPUTS:

```

```

!   Argument List: None
!   Common: IGR, ITPA, LVLP, NFACS, NROUT, NZOUT, PEFLAG, TROPO

! OUTPUTS:
!   Argument List: IERROR
!   Common: None
!   Public: ADIF(), CN2(), D2S(), DIELEC(,), FSL(), GRDUM(), HFANGR(), HLIM(),
!           HREF(), HTDUM(), HTFE(), IGRND(), RDT(), REFDUM(), REFREF(),
RFAC1(),
!           RFAC2(), RGRND(), RLOGO(), RLOSS(), RNGOUT(), RSQRD(), SLP(),
THETA0(),
!           THETA2S(), TX(), TY(), ZOUT(), ZOUTMA(), ZOUTPA()

! Modules Used: APM_MOD

! Calling Routines: APMINIT

! Routines called:
!   APM Specific: NONE
!   Intrinsic: ALLOCATE, ALLOCATED, CMLPX, DEALLOCATE

! GLOSSARY: See universal glossary for common and public variables.

!   Input Variables: None

!   Output Variables:
!           IERROR = Integer variable indicating error # for DEALLOCATE and ALLOCATE
!                   statements.

subroutine allarray_apm( IERROR )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

ierror = 0

if( nfacs .gt. 0 ) then
  IF( ALLOCATED( HFANGR ) ) DEALLOCATE( HFANGR, stat=ierror )
  ALLOCATE( HFANGR(NFACS), stat=ierror )
  if( ierror .ne. 0 ) return
  HFANGR = 0.d0
end if

if( allocated( fsl ) ) deallocate( fsl, stat=ierror )
allocate( fsl(nROUT), stat=ierror )
if( ierror .ne. 0 ) return
fsl = 0.d0

if( allocated( rlogo ) ) deallocate( rlogo, stat=ierror )
allocate( rlogo(nROUT), stat=ierror )
if( ierror .ne. 0 ) return
rlogo = 0.d0

if( allocated( rngout ) ) deallocate( rngout, stat=ierror )
allocate( rngout(nROUT), stat=ierror )
if( ierror .ne. 0 ) return
rngout = 0.d0

if( allocated( zout ) ) deallocate( zout, stat=ierror )
allocate( zout(0:nzout), stat=ierror )
if( ierror .ne. 0 ) return
zout = 0.d0

if( .not. peflag ) then
  if( allocated( rsqrd ) ) deallocate( rsqrd, stat=ierror )
  allocate( rsqrd(nROUT), stat=ierror )
  if( ierror .ne. 0 ) return
  rsqrd = 0.d0

```

```

    if( allocated( zoutma ) ) deallocate( zoutma, stat=ierror )
    allocate( zoutma(0:nzout), stat=ierror )
    if( ierror .ne. 0 ) return
    zoutma = 0.d0

    if( allocated( zoutpa ) ) deallocate( zoutpa, stat=ierror )
    allocate( zoutpa(0:nzout), stat=ierror )
    if( ierror .ne. 0 ) return
    zoutpa = 0.d0

    if( allocated( hlim ) ) deallocate( hlim, stat=ierror )
    allocate( hlim(nrout), stat=ierror )
    if( ierror .ne. 0 ) return
    hlim = 0.d0

    if( allocated( htfe ) ) deallocate( htfe, stat=ierror )
    allocate( htfe(nrout), stat=ierror )
    if( ierror .ne. 0 ) return
    htfe = 0.d0
end if

if( allocated( rfac1 ) ) deallocate( rfac1, stat=ierror )
allocate( rfac1(0:nzout), stat=ierror )
if( ierror .ne. 0 ) return
rfac1 = 0.d0

if( allocated( rfac2 ) ) deallocate( rfac2, stat=ierror )
allocate( rfac2(0:nzout), stat=ierror )
if( ierror .ne. 0 ) return
rfac2 = 0.d0

if( allocated( rloss ) ) deallocate( rloss, stat=ierror )
allocate( rloss(0:nzout), stat=ierror )
if( ierror .ne. 0 ) return
rloss = 0.d0

! Allocate arrays associated with terrain info.

if( allocated( tx ) ) deallocate( tx, stat=ierror )
allocate( tx(itpa), stat=ierror )
if( ierror .ne. 0 ) return
tx = 0.d0

if( allocated( ty ) ) deallocate( ty, stat=ierror )
allocate( ty(itpa), stat=ierror )
if( ierror .ne. 0 ) return
ty = 0.d0

if( allocated( slp ) ) deallocate( slp, stat=ierror )
allocate( slp(itpa), stat=ierror )
if( ierror .ne. 0 ) return
slp = 0.d0

if( igr .eq. 0 ) then
    igr = 1
    IF( ALLOCATED( DIELEC ) ) DEALLOCATE( DIELEC, stat=ierror )
    ALLOCATE( DIELEC(2, IGR), stat=ierror )
    if( ierror .ne. 0 ) return
    DIELEC = 0.d0

    IF( ALLOCATED( IGRND ) ) DEALLOCATE( IGRND, stat=ierror )
    ALLOCATE( IGRND(IGR), stat=ierror )
    if( ierror .ne. 0 ) return
    IGRND = 0

    IF( ALLOCATED( RGRND ) ) DEALLOCATE( RGRND, stat=ierror )
    ALLOCATE( RGRND(IGR), stat=ierror )
    if( ierror .ne. 0 ) return
    RGRND = 0.d0
end if

```

```

IF( ALLOCATED( cn2 ) ) DEALLOCATE( cn2, stat=ierror )
ALLOCATE( cn2(IGR), stat=ierror )
if( ierror .ne. 0 ) return
cn2 = cmplx(0., 0., 8)

! Allocate arrays associated with refractivity info.

if( allocated( refdum ) ) deallocate( refdum, stat=ierror )
allocate( refdum(0:lvlp), stat=ierror )
if( ierror .ne. 0 ) return
refdum = 0.d0

if( allocated( htdum ) ) deallocate( htdum, stat=ierror )
allocate( htdum(0:lvlp), stat=ierror )
if( ierror .ne. 0 ) return
htdum = 0.d0

if( allocated( grdum ) ) deallocate( grdum, stat=ierror )
allocate( grdum(0:lvlp), stat=ierror )
if( ierror .ne. 0 ) return
grdum = 0.d0

if( allocated( href ) ) deallocate( href, stat=ierror )
allocate( href(0:lvlp), stat=ierror )
if( ierror .ne. 0 ) return
href = 0.d0

if( allocated( refref ) ) deallocate( refref, stat=ierror )
allocate( refref(0:lvlp), stat=ierror )
if( ierror .ne. 0 ) return
refref = 0.d0

!Initialize and allocate arrays associated with troposcatter calculations.

if( tropo ) then

    if( allocated( adif ) ) deallocate( adif, stat=ierror )
    allocate( adif(0:nzout), stat=ierror )
    if( ierror .ne. 0 ) return
    adif = 0.d0

    if( allocated( d2s ) ) deallocate( d2s, stat=ierror )
    allocate( d2s(0:nzout), stat=ierror )
    if( ierror .ne. 0 ) return
    d2s = 0.d0

    if( allocated( rdt ) ) deallocate( rdt, stat=ierror )
    allocate( rdt(0:nzout), stat=ierror )
    if( ierror .ne. 0 ) return
    rdt = 0.d0

    if( allocated( theta0 ) ) deallocate( theta0, stat=ierror )
    allocate( theta0(nrout), stat=ierror )
    if( ierror .ne. 0 ) return
    theta0 = 0.d0

    if( allocated( theta2s ) ) deallocate( theta2s, stat=ierror )
    allocate( theta2s(0:nzout), stat=ierror )
    if( ierror .ne. 0 ) return
    theta2s = 0.d0

end if

end subroutine allarray_apm

```

A.1.2 Subroutine ALLARRAY_PE

```

! ***** SUBROUTINE ALLARRAY_PE *****
! Module Name: ALLARRAY_PE

```

```

! Module Security Classification: UNCLASSIFIED

! Purpose: This routine allocates and initializes all dynamically
!          dimensioned arrays associated with PE calculations.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: None
!   Common: IPOL, N, NF4, RUF

! OUTPUTS:
!   Argument List: IERROR
!   Common: None
!   Public: ENVPR(), FILT(), FRSP(), HT(), PROFINT(), RN(), U(), UDUM(),
!           ULST(), W(), YM()

! Modules Used: APM_MOD

! Calling Routines: PEINIT

! Routines called:
!   APM Specific: NONE
!   Intrinsic: ALLOCATE, ALLOCATED, CMPLX, DEALLOCATE

! GLOSSARY: See universal glossary for common and public variables.

!   Input Variables: None

!   Output Variables:
!       IERROR = Integer variable indicating error # for DEALLOCATE and ALLOCATE
!               statements.

!   Local Variables: None

subroutine allarray_pe( IERROR )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

ierror = 0

if( allocated( envpr ) ) deallocate( envpr, stat=ierror )
allocate( envpr(0:n), stat=ierror )
if( ierror .ne. 0 ) return
envpr = cmplx( 0., 0., 8 )

if( allocated( frsp ) ) deallocate( frsp, stat=ierror )
allocate( frsp(0:n), stat=ierror )
if( ierror .ne. 0 ) return
frsp = cmplx( 0., 0., 8 )

if( allocated( u ) ) deallocate( u, stat=ierror )
allocate( u(0:n), stat=ierror )
if( ierror .ne. 0 ) return
u = cmplx( 0., 0., 8 )

if( allocated( ulst ) ) deallocate( ulst, stat=ierror )
allocate( ulst(0:n), stat=ierror )
if( ierror .ne. 0 ) return
ulst = cmplx( 0., 0., 8 )

if( allocated( filt ) ) deallocate( filt, stat=ierror )
allocate( filt(0:nf4), stat=ierror )
if( ierror .ne. 0 ) return
filt = 0.d0

if( allocated( ht ) ) deallocate( ht, stat=ierror )

```



```

allocate( ht(0:n), stat=ierror )
if( ierror .ne. 0 ) return
ht = 0.d0

if( allocated( profint ) ) deallocate( profint, stat=ierror )
allocate( profint(0:n), stat=ierror )
if( ierror .ne. 0 ) return
profint = 0.d0

if( allocated( udum ) ) deallocate( udum, stat=ierror )
allocate( udum(0:n), stat=ierror )
if( ierror .ne. 0 ) return
udum = 0.d0

if( ( ipol .eq. 1 ) .or. ( ruf ) ) then

    if( allocated( rn ) ) deallocate( rn, stat=ierror )
    allocate( rn(0:n), stat=ierror )
    if( ierror .ne. 0 ) return
    rn = cmplx( 0., 0., 8 )

    if( allocated( w ) ) deallocate( w, stat=ierror )
    allocate( w(0:n), stat=ierror )
    if( ierror .ne. 0 ) return
    w = cmplx( 0., 0., 8 )

    if( allocated( ym ) ) deallocate( ym, stat=ierror )
    allocate( ym(0:n), stat=ierror )
    if( ierror .ne. 0 ) return
    ym = cmplx( 0., 0., 8 )

end if

end subroutine allarray_pe

```

A.1.3 Subroutine ALLARRAY_RO

```

! ***** SUBROUTINE ALLARRAY_RO *****

! Module Name: ALLARRAY_RO

! Module Security Classification: UNCLASSIFIED

! Purpose: This routine allocates and initializes all dynamically dimensioned
!          arrays associated with RO calculations.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: None
!   Common: LVLP, NZOUT

! OUTPUTS:
!   Argument List: IERROR
!   Common: None
!   Public: GR(), Q(), RM(), ZRO(), ZRT()

! Modules Used: APM_MOD

! Calling Routines: APMINIT

! Routines called:
!   APM Specific: NONE
!   Intrinsic: ALLOCATE, ALLOCATED, DEALLOCATE

! GLOSSARY: See universal glossary for common and public variables.

!   Input Variables: None

!   Output Variables:

```

```

!           IERROR = Integer variable indicating error # for DEALLOCATE and ALLOCATE
!           statements.

!   Local Variables:
!           LVLPT = LVLP + 1 -> upper boundary limit on arrays G, RM, Q

subroutine allarray_ro( IERROR )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

ierror = 0

if( allocated( zro ) ) deallocate( zro, stat=ierror )
allocate( zro(0:nzout), stat=ierror )
if( ierror .ne. 0 ) return
zro = 0.d0

lvlpt = lvlp + 1
if( allocated( gr ) ) deallocate( gr, stat=ierror )
allocate( gr(0:lvlpt), stat=ierror )
if( ierror .ne. 0 ) return
gr = 0.d0

if( allocated( q ) ) deallocate( q, stat=ierror )
allocate( q(0:lvlpt), stat=ierror )
if( ierror .ne. 0 ) return
q = 0.d0

if( allocated( rm ) ) deallocate( rm, stat=ierror )
allocate( rm(0:lvlpt), stat=ierror )
if( ierror .ne. 0 ) return
rm = 0.d0

if( allocated( zrt ) ) deallocate( zrt, stat=ierror )
allocate( zrt(0:lvlpt), stat=ierror )
if( ierror .ne. 0 ) return
zrt = 0.d0

end subroutine allarray_ro

```

A.1.4 Subroutine ALLARRAY_XORUF

```

!***** SUBROUTINE ALLARRAY_XORUF *****

! Module Name: ALLARRAY_XORUF

! Module Security Classification: UNCLASSIFIED

! Purpose: This routine allocates and initializes all dynamically
!          dimensioned arrays associated with XO and/or rough surface
!          calculations.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: None
!   Common: IX0, IZMAX, LVLP, NP4, NROUT, NS

! OUTPUTS:
!   Argument List: IERROR
!   Common: None
!   Public: FFACZ(,), FFROUT(,), FILTP(), GRAD(), HTR(), LVL(), SPECTR(),
!           XP(), YP()

! Modules Used: APM_MOD

```

```

! Calling Routines: APMINIT

! Routines called:
!   APM Specific: NONE
!   Intrinsic: ALLOCATE, ALLOCATED, DEALLOCATE

! GLOSSARY: See universal glossary for common and public variables.

!   Input Variables: None

!   Output Variables:
!       IERROR = Integer variable indicating error # for DEALLOCATE and ALLOCATE
!               statements.

!   Local Variables: None

subroutine allarray_xoruf( IERROR )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

ierror = 0

if( ixo .gt. 0 ) then
    if( allocated( ffROUT ) ) deallocate( ffROUT, stat=ierror )
    allocate( ffROUT(nrout,2), stat=ierror )
    if( ierror .ne. 0 ) return
    ffROUT = 0.d0

    if( allocated( ffacz ) ) deallocate( ffacz, stat=ierror )
    allocate( ffacz(izmax,3), stat=ierror )
    if( ierror .ne. 0 ) return
    ffacz = 0.d0

    if( allocated( grad ) ) deallocate( grad, stat=ierror )
    allocate( grad(0:lvlp,izmax), stat=ierror )
    if( ierror .ne. 0 ) return
    grad = 0.d0

    if( allocated( htr ) ) deallocate( htr, stat=ierror )
    allocate( htr(0:lvlp,izmax), stat=ierror )
    if( ierror .ne. 0 ) return
    htr = 0.d0

    if( allocated( lvl ) ) deallocate( lvl, stat=ierror )
    allocate( lvl(izmax), stat=ierror )
    if( ierror .ne. 0 ) return
    lvl = 0.d0
end if

! Allocate and initialize all arrays associated with spectral estimation
! of PE field.

if( allocated( filtp ) ) deallocate( filtp, stat=ierror )
allocate( filtp(0:np4), stat=ierror )
if( ierror .ne. 0 ) return
filtp = 0.d0

if( allocated( xp ) ) deallocate( xp, stat=ierror )
allocate( xp(0:ns), stat=ierror )
if( ierror .ne. 0 ) return
xp = 0.d0

if( allocated( yp ) ) deallocate( yp, stat=ierror )
allocate( yp(0:ns), stat=ierror )
if( ierror .ne. 0 ) return
yp = 0.d0

if( allocated( spectr ) ) deallocate( spectr, stat=ierror )

```

```

allocate( spectr(0:ns), stat=ierror )
if( ierror .ne. 0 ) return
spectr = 0.d0

end subroutine allarray_xoruf

```

A.1.5 Subroutine ALN_INIT

```

!***** SUBROUTINE ALN_INIT *****

! Module Name: ALN_INIT

! Module Security Classification: UNCLASSIFIED

! Purpose: This routine initializes variables used in DMFT algorithm for
!          finite conductivity and/or rough surface calculations.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: ASTA, RS
!   Common: IALG, NM1, RK
!   Public: RN(), U()

! OUTPUTS:
!   Argument List: None
!   Common: CK1, CK2, CMFT

! Modules Used: APM_MOD

! Calling Routines: APMINIT

! Routines called:
!   APM Specific: GETALN
!   Intrinsic: MOD, SUM

! GLOSSARY: See universal glossary for common and public variables.

!   Input Variables:
!     ASTA = Grazing angle.
!     RS = Current range.

!   Output Variables: None

!   Local Variables:
!     C2C = Running multiplicative factor for U * RN summation.

subroutine aln_init( asta, rs )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

complex(kind=8) c2c

call getaln( asta, rs )

if( ialg .eq. 1 ) then      !Use old central-difference algorithm

  ck1 = .5 * ( u(0) + u(n)*rn(n) )
  ck2 = .5 * ( u(0)*rn(n) + u(n) )

  ck1 = ck1 + sum( u(1:nm1) * rn(1:nm1) )

  do i = 1, nm1
    c2c = u(n-i) * rn(i)
    if( mod(i,2) .gt. 0 ) c2c = -c2c
    ck2 = ck2 + c2c
  end do

```

```

        ck1 = ck1 * rk
        ck2 = ck2 * rk

    else                                     !Use backward-difference algorithm

        if( ialg .eq. 2 ) cmft = sum( u(0:nm1) * rn(0:nm1) )

    end if

end subroutine aln_init

```

A.1.6 Subroutine ANTPAT

```

! ***** SUBROUTINE ANTPAT *****

! Module Name: ANTPAT

! Module Security Classification: UNCLASSIFIED

! Purpose: Determines the antenna pattern factor for angle passed to routine.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: ANG
!   Common: AFAC, ALPHAD, BW, ELV, IPAT, NFACS, PELEV, SBW, UMAX
!   Public: HFANGR(), HFFAC()

! OUTPUTS:
!   Argument List: PATFAC
!   Common: NONE

! Modules Used: APM_MOD

! Calling Routines: AIRBORNE, FEM, ROCALC, TROPOINIT, TROPOSCAT, XYINIT

! Routines called:
!   APM Specific: NONE
!   Intrinsic: DABS, DEXP, DMAX1, DSIN

! GLOSSARY: See universal glossary for common variables.

!   Input Variables:
!       ANG = elevation angle at transmitter

!   Output Variables:
!       PATFAC = antenna pattern factor for angle ANG

!   Local Variables:
!       ANGFRAC = Fractional angle used for interpolation on user-specified
!               antenna pattern.
!       CHI = Running track of angle used to compare cutback angles specified
!            for height-finder antenna pattern.
!       UDIF = Angle relative to the elevation angle of the main beam.

subroutine antpat( ang, PATFAC )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

! IPAT = 1 gives Omnidirectional antenna pattern factor : f(u) = 1
! Default for Omni antenna pattern

patfac = 1.d0

! In the following pattern definitions, "ua" refers to the angle for which
! the antenna pattern is sought, and "u0" refers to the elevation angle.

```

```

select case( ipat )

  case( 2 )

! IPAT = 2 gives Gaussian antenna pattern based on
!  $f(p-p_0) = \exp(-w^{*2} * (p-p_0)^{**2}) / 4$ , where  $p = \sin(ua)$  and
!  $p_0 = \sin(u_0)$ 

    pr = dsin(ang) - pelev
    patfac = dexp(-pr * pr * afac)

  case( 4 )

! IPAT = 4 gives csc-sq pattern based on
!  $f(u) = 1$  for  $ua-u_0 \leq bw$ 
!  $f(u) = \sin(bw) / \sin(ua-u_0)$  for  $ua-u_0 > bw$ 
!  $f(u) = \text{maximum of } 0.03 \text{ or } [1+(ua-u_0)/bw]$  for  $ua-u_0 < 0$ 

    udif = ang - elv
    if( udif .gt. bw ) then
      patfac = sbw / dsin( udif )
    elseif( udif .lt. 0 ) then
      patfac = dmax1( 0.03d0, 1.d0 + udif/bw )
    end if

  case( 3, 5, 6 )

! IPAT = 3 gives  $\sin(x)/x$  pattern based on
!  $f(ua-u_0) = \sin(x) / x$  where  $x = afac * \sin(ua-u_0)$  for  $|ua-u_0| \leq umax$ 
!  $f(ua-u_0) = .03$  for  $|ua-u_0| > umax$ 
! IPAT = 5 gives height-finder pattern which is a special case of  $\sin(x)/x$ 

    udif = ang - elv
    if( ipat .ge. 5 ) then
      chi = elv
      if( alphad .gt. elv ) then
        udif = ang - alphad
        chi = alphad
      end if
    end if

    if( dabs(udif) .le. 1.d-6 ) then
      patfac = 1.d0
    elseif( dabs( udif ) .gt. umax ) then
      patfac = 0.03d0
    else
      arg = afac * dsin( udif )
      patfac = dsin( arg ) / arg
    end if

! For IPAT = 6, user-specified height-finder antenna pattern,
! adjust user-defined height-finder pattern by appropriate factor
! based on HFANGR() and HFFAC() arrays. Adjustment is not necessary
! when CHI is less than the first user-defined angle (HFANGR(1)).

    IF( ipat .EQ. 6 ) then
      if( chi .GT. hfangr(1) ) THEN
        i = nfacs
        DO WHILE (chi .LE. hfangr(i))
          i = i - 1
        END DO
        patfac = patfac * hffac(i)
      end if
    END IF

  case( 7 )

! For IPAT=7, the antenna pattern factor is determined by linear interpolation
! on the user-specified antenna patter factor and angle arrays defined by
! HFANG() and HFFAC().

```

```

    angl = ang - elv
    if( angl .ge. hfangr(nfacs) ) then
        patfac = hffac(nfacs)
    elseif( angl .le. hfangr(1) ) then
        patfac = hffac(1)
    else
        i = 1
        do while( angl .ge. hfangr(i) )
            i = i + 1
        end do
        angfrac = (angl - hfangr(i-1)) / (hfangr(i) - hfangr(i-1))
        patfac = hffac(i-1) + angfrac * (hffac(i) - hffac(i-1))
    end if

    case default

        ! do nothing

end select

end subroutine antpat

```

A.1.7 Subroutine DIEINIT

```

! ***** SUBROUTINE DIEINIT *****

! Module Name: DIEINIT

! Module Security Classification: UNCLASSIFIED

! Purpose: This routine calculates conductivity and permittivity
!          as a function of frequency in MHz. All equations and coef-
!          ficients were obtained by using a SUMMASKETCH digitizer to digitize
!          the CCIR volume 5 curves on page 74. The digitized data were
!          then used with TABLECURVE software to obtain the best fit
!          equations and coefficients used in this subroutine. In some
!          cases two sets of equations were required to obtain a decent
!          fit across the 100 MHz - 100GHz range. These curves fit the
!          digitized data to within 5%.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: NONE
!   Common:  FREQ, IGR, WL
!   Public:  DIELEC(,), IGRND(), RGRND()

! OUTPUTS:
!   Argument List: NONE
!   Public:  CN2()

! Modules Used:  APM_MOD

! Calling Routines:  APMINIT

! Routines called:
!   APM Specified:  NONE
!   Intrinsic:  CMPLX, DSQRT

! GLOSSARY:

!   Input Variables:
!       See universal glossary for common and public variables

!   Output Variables:
!       See universal glossary for common and public variables

!   Local Variables:
!       EPSILON = relative permittivity

```

```

!      SIGMA = conductivity
!      F1-8 = Frequency in MHz to the nth power. i.e., f5 = freq**5
!      A() thru F() = polynomial coefficients for use in determining
!      EPSILON and SIGMA.

subroutine dieinit

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

dimension a(18), b(18), c(18), d(18), e(18), f(18)

data (a(i),i=1,18) / 1.4114535d-2, 3.8586749d0, 79.027635d0,      &
-0.65750351d0, 201.97103d0, 857.94335d0,      &
915.31026d0, 0.8756665d0, 5.5990969d-3,      &
215.87521d0, .17381269d0, 2.4625032d-2,      &
-4.9560275d-2, 2.2953743d-4, .000038814567d0, &
1.2434792d-04, 51852.543d0, 4.13105d-05 /
data (b(i),i=1,18) / -5.2122497d-8, -2.1179295d-5, -2.2083308d-5, &
5.5620223d-5, -2.5539582d-3, -8.9983662d-5, &
-9.4530022d-6, 4.7236085d-5, 8.7798277d-5, &
-7.6649237d-5, 1.2655183d-4, 1.8254018d-4, &
2.9876572d-5, -8.1212741d-7, 8.467523d-02, &
2.824598d-04, 3.883854d-02, 2.03589d-07 /
data (c(i),i=1,18) / 5.8547829d-11, 9.1253873d-4, -3.5486605d-4, &
6.6113198d-4, 1.2197967d-2, 5.5275278d-2, &
-4.0348211d-3, 2.6051966d-8, 6.2451017d-8, &
-2.6151055d-3, -1.6790756d-9, -2.664754d-8, &
-3.0561848d-10, 1.8045461d-9, 9.878241d-06, &
8.680839d-07, 389.58894d0, -3.1739d-12 /
data (d(i),i=1,18) / -7.6717423d-16, 6.5727504d-10, 2.7067836d-9, &
3.0140816d-10, 3.7853169d-5, 8.8247139d-8, &
4.892281d-8, -9.235936d-13, -7.1317207d-12, &
1.2565999d-8, 1.1037608d-14, 7.6508732d-12, &
1.1131828d-15, -1.960677d-12, -9.736703d-05, &
-6.755389d-08, 6.832108d-05, 4.52331d-17 /
data (e(i),i=1,18) / 2.9856318d-21, 1.5309921d-8, 8.210184d-9, &
1.4876952d-9, -1.728776d-6, 0.d0, &
7.4342897d-7, 1.4560078d-17, 4.2515914d-16, &
1.9484482d-7, -2.9223433d-20, -7.4193268d-16, &
0.d0, 1.2569594d-15, 7.990284d-08, &
7.2701689d-11, 0.d0, 0.d0 /
data (f(i),i=1,18) / 0.d0, -1.9647664d-15, -1.0007669d-14, 0.d0, 0.d0, &
0.d0, 0.d0, -1.1129348d-22, -1.240806d-20, 0.d0, &
0.d0, 0.d0, 0.d0, -4.46811d-19, 3.269059d-07, &
2.8728975d-12, 0.d0, 0.d0 /

f1 = freq
f2 = f1 * f1
f3 = f1 * f2
f4 = f1 * f3
f5 = f1 * f4
f6 = f1 * f5
f7 = f1 * f6
f8 = f1 * f7
f9 = f1 * f8

do i = 1, igr
  select case ( igrnd(i) )
    case( 0 ) ! Permittivity and conductivity for salt water
      epsilon = 70.d0
      sigma = 5.d0
      m = 1
      m1 = m + 1
      if( f1 .gt. 2253.5895d0 ) epsilon = 1.d0 / ( a(m) + b(m)*f1      &
+ c(m)*f2 + d(m)*f3 + e(m)*f4 )
      if( f1 .gt. 1106.207d0 ) then

```



```

        sigma = a(m1) + c(m1)*f1 + e(m1)*f2
        sigma = sigma / ( 1.d0 + b(m1)*f1 + d(m1)*f2 + f(m1)*f3 )
    end if

case( 1 ) !Permittivity and conductivity for fresh water
    epsilon = 80.d0
    m = 3
    m1 = m + 1
    IF( f1 .gt. 6165.776d0 ) THEN
        epsilon = a(m) + c(m)*f1 + e(m)*f2
        epsilon = epsilon/(1.d0 + b(m)*f1 + d(m)*f2 + f(m)*f3 )
    end if
    IF( f1 .gt. 5776.157d0 ) THEN
        k = 2
    else
        m1 = m1 + 1
        k = -1
    end if
    sigma = a(m1) + c(m1)*f1 + e(m1)*f2
    sigma = (sigma / (1.d0 + b(m1)*f1 + d(m1)*f2))**k

case( 2 ) !Permittivity and conductivity for wet ground
    epsilon = 30.d0
    m = 6
    IF( f1 .ge. 4228.11d0 ) m = 7
    if( f1 .gt. 1312.054d0 ) then
        epsilon = a(m) + c(m)*f1 + e(m)*f2
        epsilon = dSqrt( epsilon / (1.d0 + b(m)*f1 + d(m)*f2) )
    end if
    IF( f1 .gt. 15454.4d0 ) then
        m1 = 8
        g = 3.3253339d-28
    else
        m1 = 9
        g = 1.3854354d-25
    end if
    sigma = a(m1) + b(m1)*f1 + c(m1)*f2 + d(m1)*f3 + e(m1)*f4
    sigma = sigma + f(m1)*f5 + g*f6

case( 3 ) !Permittivity and conductivity for medium dry ground
    epsilon = 15.d0
    IF( f1 .gt. 4841.945d0 ) THEN
        m = 10
        epsilon = a(m) + c(m)*f1 + e(m)*f2
        epsilon = dSqrt( epsilon / (1.d0 + b(m)*f1 + d(m)*f2) )
    end if
    m1 = 12
    IF( f1 .gt. 4946.751d0 ) m1 = 11
    sigma = (a(m1) + b(m1)*f1 + c(m1)*f2 + d(m1)*f3 + e(m1)*f4)**2

case( 4 ) !Permittivity and conductivity for very dry ground
    epsilon = 3.d0
    IF( f1 .lt. 590.8924d0 ) then
        sigma = 1.0d-4
    else
        IF( f1 .gt. 7131.933d0 ) THEN
            m1 = 13
            sigma = (a(m1) + b(m1)*f1 + c(m1)*f2 + d(m1)*f3)**2
        else
            m1 = 14
            g = 9.4623158d-23
            h = -1.1787443d-26
            s = 7.9254217d-31
            t = -2.2088286d-35
            sigma = a(m1) + b(m1)*f1 + c(m1)*f2 + d(m1)*f3
            sigma = sigma + e(m1)*f4 + f(m1)*f5 + g*f6
            sigma = sigma + h*f7 + s*f8 + t*f9
        end if
    end if

case( 5 ) !Permittivity and conductivity for ice at -1 degree C

```

```

epsilon = 3.d0
IF( f1 .le. 300.d0 ) THEN
  m = 15
  signum = a(m) + c(m) * f1 + e(m) * f2
  sigdnom = 1.d0 + b(m) * f1 + d(m) * f2 + f(m) * f3
ELSE
  m = 16
  g = -2.6416983d-14
  h = -1.8795958d-18
  si = 1.37552d-18
  signum = a(m) + c(m)*f1 + e(m)*f2 + g*f3 + si*f4
  sigdnom = 1.d0 + b(m)*f1 + d(m)*f2 + f(m)*f3 + h*f4
END IF
sigma = signum / sigdnom

case( 6 ) !Permittivity and conductivity for ice at -10 degrees C
epsilon = 3.d0
IF( f1 .le. 8753.398d0 ) THEN
  m = 17
  sigma = 1.d0 / ((a(m) + c(m)*f1) / (1.0 + b(m)*f1 + d(m)*f2))
ELSE
  m = 18
  sigma = a(m) + b(m)*f1 + c(m)*f2 + d(m)*f3
END IF

case( 7 )
epsilon = dielec(1,i)
sigma = dielec(2,i)

case default
! Do nothing
end select

s1 = sigma * 60.d0 * wl
cn2(i) = cmplx( epsilon, s1, 8 )

end do

end subroutine dieinit

```

A.1.8 Subroutine FFTPAR

```

! ***** SUBROUTINE FFTPAR *****

! Module Name: FFTPAR

! Module Security Classification: UNCLASSIFIED

! Purpose: Determines and computes the FFT size needed for a given
!          problem, plus all other associated PE variables.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: IFLAG, LNMIN, THETAMAX, WL
!   Common: NONE

! OUTPUTS:
!   Argument List: DELZ, IERR, LN, N, ZLIM, ZMAX
!   Common: NONE

! Calling Routines: APMINIT, GETTHMAX

! Routines Called:
!   APM Specific: NONE
!   Intrinsic: DBLE, DSIN

! GLOSSARY:

!   Input Variables:

```

```

!      IFLAG = flag indicating whether to determine maximum FFT size
!              based on given THETAMAX and height needed to reach (ZLIM),
!              or determine maximum height ZLIM based on given THETAMAX
!              and FFT size.
!              = 0 -> determine N, LN given THETAMAX and ZLIM
!              = 1 -> determine ZLIM given THETAMAX and LN
!      LNMIN = Minimum power of 2 transform size.
!      THETAMAX = Maximum PE propagation angle in radians.
!      WL = Wavelength in meters

!      Output Variables:
!      DELZ = Bin width in z-space = WL / (2*sin(THETAMAX))
!      IERR = Integer error flag indicating of transform size computed
!              exceeds integer value limit of 2**31 - 1.
!      LN = Power of 2 transform size, i.e. N = 2**LN
!      N = Transform size
!      ZLIM = Maximum internal height (HTLIM) or .75*ZMAX, whichever is smaller.
!      ZMAX = Maximum height of PE calculation domain = N * DELZ

!      Local Variables:
!      STHETAMAX = Sine of THETAMAX.

subroutine fftpar( lnmin, wl, thetamax, iflag, ZLIM, ZMAX, DELZ, LN, N, IERR )

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

sthetamax = dsin( thetamax )
delz= wl * .5 / sthetamax

if( iflag .eq. 0 ) then
! Lower FFT limit is 2**LNMIN.

    ln = lnmin
    N=2**LN
    zmax=delz*dble(n)

! Determine transform size needed to perform calculations to a height of ZLIM.
! If transform size computed is at least to 2**30 then return error.

    do while( ( .75*zmax .lt. (zlim - zlim*1.d-5) ) .and. ( ierr .eq. 0 ) )
        ln = ln + 1
        if( ln .eq. 30 ) then
            ierr = -43
            return
        else
            n = 2**ln
            zmax = delz * dble(n)
        end if
    end do

elseif( iflag .eq. 1 ) then

! Determine the maximum height that can be reached given THETAMAX
! and LN.

    N=2**LN
    zmax=delz*dble(n)

end if

zlim = .75 * zmax

end subroutine fftpar

```

A.1.9 Subroutine FILLHT

```

! ***** SUBROUTINE FILLHT *****

```

```

! Module Name: FILLHT

! Module Security Classification: UNCLASSIFIED

! Purpose: Fills the array HTFE() with height values of the limiting
!           sub-model (depending on value of IHYBRID) at each output range.
!           I.e., if IHYBRID = 1, then HTFE() will contain height values at
!           each output range separating the FE region from the R0 region.
!           If IHYBRID = 0 or 2, then HTFE() will contain those height
!           values at each output range at which the initial launch angle
!           has been traced to the ground or surface. These height values
!           represent the separating region where, above that height, valid
!           loss is computed, and below that height, 1) for IHYBRID=2, no loss
!           is computed (outside PE angle limit; or 2) for IHYBRID=0, this
!           separates the lower FE region from the PE region.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: NONE
!   Common: ALAUNCH, ANTREF, DR, FTER, HMREF, HTLIM, IHYBRID, ISTART1, LVLEP
!           NROUT, THETA75, YFREF
!   Public: GRDUM(), HTDUM(), RNGOUT(), TYH()
!   Data: DEG5, RTST, TAN5

! OUTPUTS:
!   Argument List: IHMX
!   Public: HLIM(), HTFE()

! Modules Used: APM_MOD

! Calling Routines: APMINIT

! Routines Called:
!   APM Specific: PLINT, TRACE_ROUT
!   Intrinsic: MAX0, MIN0, DBLE, DMAX1, DMIN1, DSQRT, IDINT

! GLOSSARY:

!   Input Variables: See universal glossary for common variables.

!   Output Variables: See universal glossary for common variables.

!   Local Variables:
!     A0 = Angle at start of trace in radians.
!     A1 = Angle at end of trace in radians.
!     GRD = Gradient of current refractivity layer being traced through.
!     H0 = Height at start of trace in meters.
!     H1 = Height at end of trace in meters.
!     H5 = Height of 5 degree ray traced to each output range point
!     IQUIT = Flag to end loop.
!     JL = Index in indicating level in refractivity profile location of
!           source height ANTREF.
!     R0 = Range at start of trace in meters.
!     R1 = Range at end of trace in meters.
!     R0 = Current output range to trace to.
!     YAR = Height of image source.

subroutine fillht

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

! Define in-line ray trace functions:

rada1( a, b ) = a**2 + 2. * grd * b           !a=a0, b=h1-h0
rp( a, b ) = a + b / grd                      !a=r0, b=a1-a0
ap( a, b ) = a + b * grd                      !a=a0, b=r1-r0
hp( a, b, c ) = a + ( b**2 - c**2 ) / 2. / grd !a=h0, b=a1, c=a0

```

```

if( ihybrid .eq. 1 ) then
! Fill height array separating FE region from R0 region.

  yar = yfref - antref
  do i = 1, nrout
    htfe(i) = yfref
    if( rngout(i) .gt. rtst ) then
      h5 = dmax1( yfref, yar + tan5 * rngout(i) )
      htfe(i) = dmin1( htlim, h5 )
    end if
  end do

elseif( ihybrid .eq. 0 ) then          !Airborne hybrid model
! Fill height array separating upper FE region from PE region [HLIM()].

  call trace_rout( deg5, 0.d0, antref, istart1, nrout, htlim, grdum, htdum, &
    lvlep, rngout, IHMX, HLIM )

end if

if( ihybrid .ne. 1 ) then

! For PE+X0 running mode or airborne hybrid model, trace initial launch
! angle until it hits ground, storing heights traced at each output range.

  a0 = -alaunch
  if( ihybrid .eq. 0 ) a0 = -theta75
  r0 = 0.
  h0 = antref
  jl = istart1
  iquit = 0
  jr = 1
  tyh_r = 0.
  iref = 0

  do while(( iquit .eq. 0 ) .and. ( jr .le. nrout ))
    htfe(jr) = 0.
    ro = rngout(jr)

    do while(( r0 .lt. ro ) .and. ( iref .eq. 0 ))

      r1 = ro
      grd = grdum(jl)
      a1 = ap( a0, r1-r0 )
      h1 = hp( h0, a1, a0 )

      if( fter ) then
        isg = idint( r1 / dr )
        xx = ( r1 - dbple(isg)*dr ) / dr
        isgp1 = min0( isg + 1, ipe )
        tyh_r = plint( tyh(isg), tyh(isgp1), xx )
      end if

      if( a1 .lt. 0. ) then
        if(( h1 .le. (tyh_r + h1*1.d-5) ) .and. ( tyh_r .ge. htdum(jl)) ) then
          h1 = tyh_r
          rad = rada1( a0, h1-h0 )
          a1 = -dsqrt( rad )
          r1 = rp( r0, a1-a0 )
          iref = 1
        end if
        if( h1 .lt. (htdum(jl) + h1*1.d-5) ) then
          h1 = dmax1( htdum(jl), htdum(0) )
          rad = rada1( a0, h1-h0 )
          a1 = -dsqrt( rad )
          r1 = rp( r0, a1-a0 )
          jl = max0( 0, jl - 1 )
        elseif( h1 .lt. htdum(0) + h1*1.d-5 ) then

```

```

        h1 = hmref
        rad = rada1( a0, h1-h0 )
        a1 = -dsqrt( rad )
        r1 = rp( r0, a1-a0 )
        iref = 1
    end if
else
    iquit = 1
end if

    a0 = a1
    h0 = h1
    r0 = r1

end do

    htfe(jr) = h0
    if( iref .eq. 1 ) iquit = 1
    jr = jr + 1

end do

jr = jr - 1
if( iref .eq. 1 ) htfe(jr:nrout) = hmref

end if

end subroutine fillht

```

A.1.10 Subroutine GASABS

```

! ***** SUBROUTINE GASABS *****
!
! Module Name: GASABS
!
! Module Security Classification: UNCLASSIFIED
!
! PURPOSE: Computes sea-level gaseous absorption from temperature,
!          absolute humidity, and radio frequency using CCIR
!          Recommendation 676-1. This routine is good for frequencies
!          less than 57 GHz, air temperature from -20 to 40 degrees C,
!          and absolute humidity from 0 to 50 g/m3.
!
! Version Number: 1.3.0
!
! INPUTS:
!   Argument List: NONE
!   Common: ABSHUM, FREQ, TAIR
!
! OUTPUTS:
!   Argument List: NONE
!   Common: GASATT
!
! Modules Used: APM_MOD
!
! Calling Routines: APMINIT
!
! Routines called: NONE
!
! GLOSSARY:
!
!   Input Variables: See universal glossary for common variables.
!
!   Output Variables: See universal glossary for common variables.
!
!   Local Variables:
!     FGHZ = Frequency in GHz.
!     FGHZ2 = Square of frequency in GHz.
!     GAMMAO = oxygen absorption in dB/km.
!     GAMMAW = water vapor absorption in dB/km.

```

```

SUBROUTINE gasabs

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

fghz = freq / 1.0d3
fghz2 = fghz * fghz

! Compute oxygen absorption for 15 degrees C air temperature.

t1 = 6.09 / (fghz2 + 0.227)
t2 = 4.81 / ((fghz - 57.0) ** 2 + 1.50)
gammao = (7.19E-3 + t1 + t2) * fghz2 * 1.0d-3

! Correct oxygen absorption for actual air temperature.

gammao = (1.0 - 0.01 * (Tair - 15.0)) * gammao

! Compute water vapor absorption.

t1 = 3.6 / ((fghz - 22.2) ** 2 + 8.5)
t2 = 10.6 / ((fghz - 183.3) ** 2 + 9.0)
t3 = 8.9 / ((fghz - 325.4) ** 2 + 26.3)
gammaw = (0.05 + 0.0021 * abshum + t1 + t2 + t3) * &
          fghz2 * abshum * 1.0d-4

! Compute total specific absorption for sea-level air in dB/m.

gasatt = (gammao + gammaw) * 1.d-3 ! Conversion for range in m.

END subroutine gasabs

```

A.1.11 Subroutine GET_K

```

!***** SUBROUTINE GET_K *****

! Purpose: This routine calculates the effective earth radius for an
!          initial launch angle of 5 degrees or the critical angle ACRIT,
!          depending on the calling routine.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: IORG
!   Common: ACRIT, ALAUNCH, ANTREF, ELV, FTER, HMREF, HTLIM, IHYBRID, ISTART1,
!           LVLEP, YFREF
!   Public: GRDUM(), HTDUM()
!   Data: DEG5, DEG10

! OUTPUTS:
!   Argument List: NONE
!   Common: AEK, EK, THETA75, TWOKA, TWOKA_DOWN

! Modules Used: APM_MOD

! Calling Routines: APMINIT, TROPPOINT

! Routines Called:
!   APM Specific: NONE
!   Intrinsic: DABS, DMAX1, DMIN1, DSQRT

! GLOSSARY:

!   Input Variables:
!       IORG = Integer flag indicating calling routine.
!       IORG = 0 -> called from APMINIT
!       IORG = 1 -> called from TROPPOINT

```

```

!   Output Variables: See universal glossary for common variables.

!   Local Variables:
!       A0 = Angle at start of trace in radians.
!       A1 = Angle at end of trace in radians.
!       ASTART = Starting launch angle for ray trace.
!       ELV_LIM = Limiting elevation angle - no more than 10 degrees.
!       GRD = Gradient of current refractivity layer being traced through.
!       H0 = Height at start of trace in meters.
!       H1 = Height at end of trace in meters.
!       HMXTMP = Maximum trace height.
!       R0 = Range at start of trace in meters.
!       R1 = Range at end of trace in meters.
!       R0 = Current output range to trace to.

subroutine get_k( iorg )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

! Define in line ray trace functions:

rada1( a, b ) = a**2 + 2. * grd * b           !a=a0, b=h1-h0
rp( a, b ) = a + b / grd                     !a=r0, b=a1-a0

astart = acrit

! Try to include as much of beamwidth and elevation angle region within PE calcs, but
! limit it to 10 degrees max.

if( iorg .eq. 0 ) then
    astart = deg5
    if( ihybrid .eq. 0 ) then                 !Airborne mode
        elv_lim = dmin1( deg10, dabs( elv ) )
        bwt = bw
        if( ipat .eq. 1 ) bwt = 0.
        astart = dmin1( dmax1( astart, bwt + elv_lim ), deg10 )
        if( fter ) astart = dmin1( astart*1.5, deg10 )
    end if
end if

if( ( ihybrid .ne. 2 ) .or. ( iorg .eq. 1 ) ) then

!Trace elevation angle from emitter height up to maximum height HTLIM.
!Then compute TWOKA (2*k*a) for portion above PE region.
!Use in routines AIRBORNE and FEM to correct heights for earth
!curvature and average refraction.  EK and AEK are used in routine TROPO.

    theta75_a = 0.
    iflag = 0
    a0 = astart
    r0 = 0.
    i = istart1
    h0 = antref
    hmxtmp = htdum(lvlep)
    DO WHILE ((htdum(i+1) .LT. hmxtmp) .AND. (i .LT. lvlep))
        grd = grdum(i)
        rad = rada1( a0, htdum(i+1)-h0 )
        a1 = dsqrt( rad )
        r1 = rp( r0, a1-a0 )
        a0 = a1
        r0 = r1
        h0 = htdum(i+1)
        if( ( h0 .gt. htlim ) .and. ( iflag .eq. 0 ) ) then
            grd = grdum(i)
            rad = rada1( a0, htlim-htdum(i) )
            a1 = dsqrt( rad )
            theta75_a = a1
        end if
    end DO

```



```

        iflag = 1
      end if
      i = i + 1
    END DO
    if(( h0 .lt. htlim ) .and. ( iflag .eq. 0 )) then
      grd = grdum(i)
      rad = rada1( a0, htlim-h0 )
      a1 = dsqrt( rad )
      theta75_a = a1
      iflag = 1
    end if
    grd = grdum(i)
    rad = rada1( a0, hmxtmp-h0 )
    a1 = dsqrt( rad )
    r1 = rp( r0, a1-a0 )

    aek = r1 / (a1 - astart)
    twoka = 2. * aek
    if( iorg .eq. 1 ) ek = aek / 6.371d6

  end if

!Trace elevation angle from emitter height down to surface.
!Then compute TWOKA_DOWN (2*k*a) for portion below PE region.
!Used in routines AIRBORNE and FEM to correct heights for earth
!curvature and average refraction.

if(( ihybrid .eq. 0 ) .and. ( iorg .eq. 0 )) then

  ala = -astart
  a1 = 0.d0
  a0 = ala
  r0 = 0.d0
  i = istart1
  h0 = antref
  DO WHILE (( h0 .ge. hmref ) .and. ( i .ge. 0 ))
    grd = grdum(i)
    rad = rada1( a0, htdum(i)-h0 )
    a1 = -dsqrt( rad )
    r1 = rp( r0, a1-a0 )
    a0 = a1
    r0 = r1
    h0 = htdum(i)
    i = i - 1
  END DO

  twoka_down = 2. * r1 / (a1 + astart)      !launch angle is negative(-5 deg.)

  theta75 = dmax1( theta75_a, astart )
end if

end subroutine get_k

```

A.1.12 Subroutine GETALN

```

! ***** SUBROUTINE GETALN *****
!
! Module Name: GETALN
!
! Module Security Classification: UNCLASSIFIED
!
! Purpose:  Computes the impedance term ALPHAQ and the complex index of
!           refraction for finite conductivity. These formulas follow
!           Dockery and Kuttler's method. Ref:
!           1) "An Improved Impedance Boundary Algorithm for Fourier Split-Step
!              Solutions of the Parabolic Wave Equation", IEEE Trans. on Ant. and
!              Prop., Vol. 44, No.12, December 1996, pp. 1592-1599.
!           2) JHU/APL in-house report A2A-00-U-0-010.
!
! Version Number: 1.3.0

```

```

! INPUTS:
!   Argument List: GRZA, R
!   Common: DELZ, DR, FK0, IALG, IG, IPOL, N, QIFKO, RUF
!   Public: CN2()
!   Data: QI

! OUTPUTS:
!   Argument List: NONE
!   Common: ALPHAQ, C1X, C2X, CMFT_X, RK, RT
!   Public: RN()

! Modules Used: APM_MOD

! Calling Routines: ALN_INIT, PESTEP

! Routines called:
!   APM Specific: GETREFCOEF
!   Intrinsic: CDEXP, CDLOG, CDSQRT, CMLPX, DSIN, REAL

! GLOSSARY:

!   Input Variables: See universal glossary for common variables.
!       GRZA = Grazing angle in radians
!       R = Current PE range in meters

!   Output Variables: See universal glossary for common variables.

!   Local Variables:
!       AD = Portion of exponent term for calculation of coefficients, C1X, C2X,
!           CMFT_X.
!       REFCOEF = Complex reflection coefficient.
!       RMAG = Magnitude of reflection coefficient.
!       RPHASE = Phase of reflection coefficient.

subroutine getaln( grza, r )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

complex(kind=8) ad, sqrad, rootln, refcoef

rk = cmlpx( 0.d0, 0.d0, 8 )

!Determine smooth surface impedance term for horizontal polarization.
if( ipol .eq. 0 ) alphaq = qifko * cdsqrt( cn2(ig) - 1.d0 )

!Determine smooth surface impedance term for vertical polarization.
if( ipol .eq. 1 ) alphaq = qifko * cdsqrt( cn2(ig) - 1.d0 ) / cn2(ig)

!Determine smooth surface impedance term for circular polarization.
if( ipol .eq. 2 ) alphaq = 2. * qifko * cdsqrt( cn2(ig) - 1.d0 ) / (cn2(ig) + 1.d0)

if(( ruf ) .and. ( grza .gt. 1.d-6 )) then                !Determine rough surface
impedance.
    call getrefcoef( 1, grza, r, REFCOEF, RMAG, RPHASE )
    alphaq = qifko * dsin(grza) * (1.d0 - refcoef ) / ( 1.d0 + refcoef )
end if

ad = alphaq * delz

if( ialg .eq. 1 ) then                !Old central difference DMFT
    sqrad = cdsqrt( 1.d0 + ad**2 )

```

```

        if( real( alphaq, 8 ) .lt. 0.d0 ) then
            rt = -sgrad - ad !H pol
        else
            rt = sgrad - ad !V pol
        end if

    else

        !Backward difference DMFT

        rt = 1.d0 / ( 1.d0 + ad )

    end if

    rn(0) = cmplx( 1.d0, 0.d0 )
    do i = 1, n
        rn(i) = rn(i-1) * rt
    end do

    rootln = cdlog( rt )
    ad = (rootln / delz)**2

    ! Compute C propagators.

    if( ialg .eq. 1 ) then
        rk = 2.*(1.d0 - rn(2)) / (1.d0 + rn(2)) / (1.d0 - rn(n)**2)
        c1x = cdexp( qi * dr * cdsqrt(fko**2 + ad) )
        ad = ( (rootln - qi * pi ) / delz )**2
        c2x = cdexp( qi * dr * cdsqrt(fko**2 + ad) )
    else
        cmft_x = cdexp( qi * dr * cdsqrt(fko**2 + ad) )
    end if

end subroutine getaln

```

A.1.13 Subroutine GETGRAZE

```

! ***** SUBROUTINE GETGRAZE *****

! Module Name: GETGRAZE

! Module Security Classification: UNCLASSIFIED

! Purpose: Computes grazing angles for use in subsequent rough surface calculations.

! Version Number 1.3.0

! INPUTS:
!   Argument List: HTRAP, THMXG
!   Common: ACUT, ANTREF, DR, DR2, FTER, IHYBRID, IPE, LDUCT, LEVAP, N, N34, NF4,
!           NPROF, PI2, RHOR, RMAX
!   Public: FILT(), TYH()
!   Parameter: PI, QI

! OUTPUTS:
!   Argument List: IERROR
!   Common: None
!   Public: GRZ_PE(), GRZ_RAY(), IGPE, IGRZ

! Modules Used: APM_MOD

! Calling Routines: APMINIT

! Routines Called:
!   APM Specific: DOSHIFT, FRSTP, RDTRACE, REFINITER, SPECEST
!   Intrinsic: ALLOCATE, ALLOCATED, CDEXP, MIN0, DMIN1, DEALLOCATE

! GLOSSARY: See universal glossary for common and public variables

!   Input Variables: See universal glossary for common variables.
!   HTRAP = Height of highest trapping layer in meters from all

```

```

!           refractivity profiles.
!           THMXG = Maximum PE propagation angle for grazing angle calcs (3 degrees).

!   Output Variables: See universal glossary for common or public variables.
!           IERROR = Integer value returned (non-zero) if errors occur in allocat-
!                   ing or deallocating GRAZE().

!   Local Variables:
!           IC = Index counter for GRZ_PE().
!           IPESTP = Integer counter for current PE range step.
!           R = Current PE range in meters.
!           RMID = Range mid-way between current and last PE range.
!           THOUT = Grazing angle determined from spectral estimation.
!           TYH_R = Height of terrain w.r.t. HMREF at each PE range step.

subroutine getgraze( htrap, thmxg, IERROR )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

ierror = 0
igpe = 0

! Perform ray trace to determine grazing angles.

call rdtrace( thmxg, IERROR )
if( ierror .ne. 0 ) return

! Compute grazing angles using PE spectral estimation only if terrain is present
! or if a surface-based duct has been specified. Determine grazing angle by complete
! PE run for smooth surface (i.e., 0. wind speed).

if( ((lduct) .and. (.not. levap)) .or. (fter) ) then

    if( allocated( grz_pe ) ) deallocate( grz_pe, stat=ierror )
    allocate( grz_pe(0:ipe), stat=ierror )
    if( ierror .ne. 0 ) return
    grz_pe = 0.d0
    grz_pe(0) = pi2

    r = 0.
    ipestp = 0
    rmid = 0.
    ic = 0
    ylast = 0.
    if( fter ) ylast = tyh(0)

    do while( r .lt. rmax )
        r = r + dr
        ipestp = min0( ipestp + 1, ipe )
        rmid = r - dr2

!Status for stand-alone APM program.
!****COMMENT THIS LINE IF INCORPORATING INTO OTHER SOFTWARE APPLICATION WITH GRAPHICS****

        write(*,*) 'Computing grazing angle for range(km) = ', r*1.e-3

        if( fter ) then
            ycur = tyh(ipestp)

! Determine height at 1/2 range step - for interpolation on refractivity
! profiles.

            ycurm = .5 * ( tyh(ipestp-1) + ycur )
            if( ycur .lt. ylast ) call doshift

        end if

! Perform free-space step advance for smooth surface, horizontal pol.

```

```

! Transform U() to p-space, then multiply by free-space propagator,
! then transform back. Upon return U() is in z-space.

      call frstp( U )

! If range-dependent case, then interpolate on profile.

      if(( nprof .gt. 1 ) .or. ( fter )) then
        call refinter( ipestp, rmid )

! Compute environmental phase term for 2nd part of PE step.

        envpr = cdexp( qi*dr * profint )

! Filter upper 1/4 of the array.

        envpr(n34:n) = filt(0:nf4) * envpr(n34:n)
      end if

! Multiply by environment term.

      u = u * envpr

      if(( fter ) .and. ( ycur .ge. ylast )) call doshift

! Determine grazing angle by spectral estimation.

      ic = min0( ic + 1, ipe )
      call specest( 1, THOUT )
      if(( .not. fter ) .and. ( r .gt. rhor ) .and. ( htrap .eq. 0. )) then
        grz_pe(ic) = dmin1( acut, thout )
      else
        grz_pe(ic) = thout
      end if
      ylast = ycur
    end do

    if( ic .lt. ipe ) grz_pe(ic:ipe) = grz_pe(ic)
    igpe = ipe

  end if

end subroutine getgraze

```

A.1.14 Subroutine GETTHMAX

```

! ***** SUBROUTINE GETTHMAX *****

! Module Name: GETTHMAX

! Module Security Classification: UNCLASSIFIED

! Purpose: This first part of this routine performs an iterative ray
!          trace such that, upon reflection, the ray clears the highest
!          terrain peak along its path by 20%. Heights and angles of
!          this ray are stored at each output range. The 2nd part of this
!          routine determines the minimum PE propagation angle
!          necessary to meet the following criteria when using the full
!          hybrid mode (i.e., IHYBRID=1): 1) Top of the PE
!          region must contain ALL trapping layers for all refractivity
!          profiles, 2) top of PE region must be at least 20% higher than
!          highest peak along terrain profile, 3) minimum PE propagation
!          angle must be at least as large as PSILIM.

! Version Number: 1.3.1

! INPUTS:
!   Argument List: ALFLIM, HTERMAX, HTEST, RFLAT, ZTEST
!   Common: ANTH, ANTREF, FREQ, FTER, HTLIM, IALG, IHYBRID, ISTART1,
!           ITPA, NROUT, RMAX, RUF, WL, YFREF, ZLIM, ZTOL

```

```

!   Data:  RADC
!   Parameter:  IRTMP
!   Public:  GRDUM(), HTDUM(), RNGOUT(), SLP(), TX(), TY()

!   OUTPUTS:
!   Argument List:  IERR, THETAMAX
!   Common:  ALAUNCH, HTEMP(), IAP, LN, NOPE, PSILIM, RAYA(), RPEST, RTEMP(),
!            THETA75, ZMAX
!   Public:  HLIM()

!   Modules Used:  APM_MOD

!   Calling Routines:  APMINIT

!   Routines Called:
!   APM Specific:  FFTPAR, TRACE_ROUT
!   Intrinsic:  DABS, DASIN, DBLE, DEXP, DMAX1, DMIN1, DSIGN, DSQRT, IDINT, MAX0, MIN0,
REAL

!   GLOSSARY:  See universal glossary for common and data variables.

!   Input Variables:
!   ALFLIM = Elevation angle of R0 limiting ray in radians. Used to
!            initialize launch angle in GETTHMAX routine.
!   HTERMAX = Maximum terrain height along profile path in meters.
!   HTEST = Minimum height in meters at which all trapping
!            refractivity features are below (includes some slop).
!   RFLAT = Maximum range in meters at which the terrain profile
!            remains flat from the source.
!   ZTEST = Height in PE region that must be reached for hybrid model.

!   Output Variables:
!   IERR = Integer flag returned from call to FFTPAR.
!   THETAMAX = Maximum propagation angle used in the PE model.

!   Local Variables:
!   A0 = Angle at start of trace in radians.
!   A1 = Angle at end of trace in radians.
!   AMLIM = Minimum PE angle limit, i.e., THETAMAX must be at least
!            this value.
!   AMXCUR = Maximum local angle along the traced ray up to ZLIM (with
!            minimum limit AMLIM).
!   ATEMP = Temporary THETAMAX, i.e., ATEMP = AMXCUR/.75
!   B1-3 = Coefficients of polynomial to determine AMLIM.
!   DRTEMP = Range step for ray tracing.
!   GRD = Gradient of current refractivity layer being traced through.
!   H0 = Height at start of trace in meters.
!   H1 = Height at end of trace in meters.
!   HMT = Used for near ranges and low antenna heights over terrain, this is a
!            multiplicative factor that lowers the 20% ray clearance limit for these
!            cases.
!   IQUIT = Integer flag indicating to quit (IQUIT=1) tracing of
!            current ray and begin again with a new launch angle.
!   IRAY = Integer flag to continue raytracing (IRAY = 0) or to stop
!            (IRAY = 1).
!   IREF = Integer flag indicating if ray has hit surface within a range step:
!            0 = no reflection
!            1 = reflection
!   JL = Index for refractivity profile - current layer being traced
!            through.
!   KT = Counter index for terrain profile arrays TX() and TY().
!   R0 = Range at start of trace in meters.
!   R1 = Range at end of trace in meters.
!   SLOPE = Slope of current terrain segment.
!   T1-3 = Constants used in polynomial to determine AMLIM.
!   TOL = Iterative height tolerance to test if appropriate combination
!            of THETAMAX, FFT size, and ZMAX has been reached to meet
!            all criteria.
!   YN = Height of terrain at current range for traced ray.
!   YNT = Height of terrain at source.
!   ZLIMT = Maximum height to trace to (includes some slop).

```

```

subroutine getthmax( htest, htermax, rflat, ztest, alflim, alim_v, THETAMAX, IERR )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

data t1, t2, t3 / 248.4, 2867., 2495. /
data b1, b2, b3 / 4.331, 1.420, .4091 /
data aoff / .37541d0 /                                !small angular offset

! Define in line ray trace functions:

rada1( a, b ) = a**2 + 2. * grd * b                    !a=a0, b=h1-h0
rp( a, b ) = a + b / grd                               !a=r0, b=a1-a0
ap( a, b ) = a + b * grd                               !a=a0, b=r1-r0
hp( a, b, c ) = a + ( b**2 - c**2 ) / 2. / grd         !a=h0, b=a1, c=a0

ierr = 0

! Determine minimum PE angle limit, AMLIM.

f1 = -freq / t1
f2 = -freq / t2
f3 = -freq / t3

i1 = 0
i2 = 0

amlim = 2. * ( aoff + b1 * dexp(f1) + b2 * dexp(f2) + b3 * dexp(f3) )
amlim = amlim * radc
if(( .not. fter ) .and. ( .not. ruf ) .and. ( ialg .eq. 0 )) i1 = 1
if(( ialg .eq. 2 ) .and. ( freq .ge. 15.d3 ) .and. ( antref .gt. 5. )) i2 = 1
if(( i1 .eq. 1 ) .or. ( i2 .eq. 1 )) amlim = .5 * amlim

!Make sure maximum angle can accommodate low antenna heights for V pol.

if( ialg .eq. 2 ) then
    alimvp = dasin( wl / 2. / antht )
    amlim = dmax1( alimvp, amlim )
end if

if( alim_v .gt. 0.) amlim = alim_v  !***FOR SPAWAR USE ONLY***

ilbef = idint(amlim / deg5 )
lnmin = lnmin + ilbef

! Determine multiplicative factor for height clearance of ray path.

den = dmax1( ty(1), 1.d0 )
hmt = dmin1( .2d0, antht / den )
if( hmt .lt. .1d0 ) hmt = 0.d0
hmt = hmt + 1.d0

!Initialize variables used in ray tracing.

alaunch = alflim
if( ihybrid .eq. 1 ) alaunch = -alaunch
zlimt = htlim - htlim*1.d-5
iray = 0
yn = 0.
ynt = 0.
hstart = 0.
if(( ihybrid .eq. 1 ) .and. ( dabs(ty(1)) .gt. 1.d-3 )) ynt = ty(1)
drtemp = rmax / dble( irtemp )

! Begin iterative ray trace to determine the launch angle, and subsequently
! THETAMAX.

do while ( iray .eq. 0 )

```

```

a0 = alaunch
r0 = 0.
h0 = antref
jl = istart1
kt = 1
slope = slp(kt)
iquit = 0
ro = 0.
iref = 0

! Begin tracing ray to each 1/IRTEMP range.

do i = 1, irtemp

  ro = ro + drtemp
  do while( r0 .lt. ro )

    r1 = ro
    grd = grdum(jl)
    a1 = ap( a0, r1-r0 )

    if( dsign(1.d0,a0) .ne. dsign(1.d0,a1) ) then
      a1 = 0.
      r1 = rp( r0, a1-a0 )
    end if
    h1 = hp( h0, a1, a0 )

    if(( a1 .ge. 0. ) .and. ( h1 .ge. (htdum(jl+1) - h1*1.d-5) )) then
      h1 = htdum(jl+1)
      rad = rada1( a0, h1-h0 )
      a1 = dsqrt( rad )
      r1 = rp( r0, a1-a0 )
      jl = jl + 1
      h1 = dmin1( htlim, htdum(jl) )
    elseif( a1 .le. 0. ) then
      if(( h1 .le. (ynt + h1*1.d-5) ) .and. ( ynt .ge. htdum(jl)) ) then
        h1 = ynt
        rad = rada1( a0, h1-h0 )
        a1 = -dsqrt( rad )
        r1 = rp( r0, a1-a0 )
        iref = 1
      end if
      if( h1 .le. (htdum(jl) + h1*1.d-5) ) then
        h1 = htdum(jl)
        rad = rada1( a0, h1-h0 )
        a1 = -dsqrt( rad )
        r1 = rp( r0, a1-a0 )
        jl = max0( 0, jl - 1 )
      end if
    end if

! The ray has hit surface and is reflected.

    if( iref .eq. 1 ) then
      a1 = -a1
      psilim = a1
      rpest = r1
      hstart = ynt
      if( r1 .gt. rflat ) iquit = 1
      if( dabs(h1-htdum(jl+1)) .le. h1*1.d-5 ) jl = jl + 1
      iref = 0
    end if

    h0 = h1
    r0 = r1
    a0 = a1
    if( iquit .eq. 1 ) exit
  end do

! Check to see that current height of ray is at least 20% higher than
! current terrain height.

```



```

    if( fter ) then
        do while((r0 .gt. tx(kt+1)) .and. (kt .lt. itpa))
            kt = kt + 1
            slope = slp(kt)
        end do
        if(( ihybrid .eq. 2 ) .and. ( r0 .le. 5.d3 )) then
            yn = hmt * (ty(kt) + slope * ( r0 - tx(kt) ))
        else
            yn = 1.2 * (ty(kt) + slope * ( r0 - tx(kt) ))
        end if
    end if

    raya(i) = a0
    htemp(i) = h0
    rtemp(i) = r0

    if( ihybrid .eq. 1 ) then
        if(( h0 .lt. yn ) .and. (r0 .gt. rflat)) iquit = 1
    else
        if(( h0 .lt. yn ) .and. ( slope .gt. 0. )) iquit = 1
    end if
    if( h0 .ge. zlimt ) exit
    if( iquit .eq. 1 ) exit

end do

! If H0 is less than current terrain height then increase (steepen) angle
! and perform ray trace again. Angle is initially downgoing for IHYBRID=1
! and upgoing otherwise.

if( iquit .eq. 1 ) then
    if( ihybrid .eq. 1 ) then
        alaunch = alaunch - 1.d-3
    else
        alaunch = alaunch + 1.d-3
    end if
else
    ! If ZLIM is reached with no problems, then set initial launch angle -
    ! ray has been found with all heights, ranges, and angles stored. Set flag
    ! to quit.

    iray = 1
    ihmax = i
end if
end do

do i = ihmax, irtemp
    htemp(i) = h0
    raya(i) = a0
    rtemp(i) = rmax
end do
ihmax = min0( ihmax, irtemp )

! Determine at what index the local ray angles become positive,
! i.e., after reflection.

j = 1
do while(( raya(j) .lt. 0. ) .and. ( j .lt. irtemp ))
    j = j + 1
end do
iap = j
if( iap .eq. irtemp ) nope = 1

if( nope .eq. 0 ) then

! Now determine THETAMAX for PE region based on local angles along ray.

    iok = 0
    iflag = 0

```

```

zlim = 0.
amxcur = 0.
tol_last = -1.

do while( iok .eq. 0 )
  j = iap
  do j = iap, ihmax
    if( htemp(j) .gt. (ztest - htemp(j)*1.d-5) ) exit
  end do
  ist = min0( j, ihmax )

  amxcur = dabs( raya(1) )
  do i = 2, ist
    if( dabs( raya(i) ) .gt. amxcur ) amxcur = raya(i)
  end do

  amxcur = dmax1( amlim, amxcur )
  atemp = amxcur / .75

  if( ihybrid .eq. 2 ) ztest = dmax1( antref, htest, 1.2*htermax, 1000. )
! Compute new ZTEST, ZMAX, DELZ, LN, N.

  call fftpar( lnmin, wl, atemp, iflag, ZTEST, ZMAX, DELZ, LN, N, IERR )
  if( ierr .ne. 0 ) return

  if( iflag .eq. 0 ) then
    iflag = 1
    if( ihybrid .ne. 1 ) iok = 1
  elseif( ( iflag .eq. 1 ) .and. ( ihybrid .ne. 2 ) ) then
    tol = dabs( ztest - zlim ) / ztest
    if( tol .le. ztol ) iok = 1
    if( ( tol-tol_last .gt. 0 ) .and. ( tol_last .gt. 0. ) ) iok = 1
    tol_last = tol
  end if
  zlim = ztest

end do

theta75 = amxcur
thetamax = atemp

ilaft = idint( theta75 / deg5 )
if( ( ln - lnmin ) .lt. ( ilaft - ilbef ) ) then
  ln = lnmin + ilaft
  N=2**LN
  zmax=delz*dble(n)
end if

!Before exiting, fill in array HLIM().

if( ihybrid .eq. 1 ) then
  call trace_rout( psilim, rpest, hstart, 0, nrout, htlim, grdum, htdum, &
    lvlep, rngout, IHMX, HLIM )
else
  call trace_rout( alaunch, 0.d0, antref, istart1, nrout, htlim, grdum, &
    htdum, lvlep, rngout, IHMX, HLIM )
end if

else
  theta75 = amlim
  thetamax = theta75 / .75
  rpest = 1.1 * rmax
  zlim = htlim
end if

do i = 1, nrout

```

```

        if( rngout(i) .lt. rpest ) hlim(i) = yfref
    end do

end subroutine getthmax

```

A.1.15 Subroutine GRAZE_INT

```

!***** SUBROUTINE GRAZE_INT *****

! Module Name: GRAZE_INT

! Module Security Classification: UNCLASSIFIED

! Purpose: Interpolates grazing angles at each PE range step based on angles
!          computed from ray trace (takes precedence) and those computed from
!          spectral estimation. These are used for subsequent rough surface
!          calculations.

! Version Number 1.3.0

! INPUTS:
!   Argument List: RFLAT
!   Common: ACUT, ANTHT, DR, DR2, DRGRZ, FTER, HMREF, IGPE, IGRZ,
!            IHYBRID, IPE, LDUCT, LEVAP, PI2, RHOR, RMAX, YFREF
!   Public: GRZ_PE(), GRZ_RAY(,)
!   Data: AEKST

! OUTPUTS:
!   Argument List: IERROR
!   Common: NONE
!   Public: GRAZE()

! Modules Used: APM_MOD

! Calling Routines: APMINIT

! Routines Called:
!   APM Specific: PLINT
!   Intrinsic: ALLOCATED, MIN0, DABS, DATAN, DBLE, DEALLOCATE, DMAX1, DMIN1, IDINT,
!              SUM

! GLOSSARY: See universal glossary for common and public variables

!   Input Variables: See universal glossary for common variables.
!       RFLAT = Maximum range in meters at which the terrain profile
!               remains flat from the source.

!   Output Variables: See universal glossary for common or public variables.
!       IERROR = Integer value returned (non-zero) if errors occur in allocat-
!               ing or deallocating GRAZE().

!   Local Variables:
!       GRZ_ANG = Interpolated grazing angle.
!       ICR = Number of grazing angles from ray trace within range interval R-DR2 to
!             R+DR2.
!       PER = Range at which grazing angles from PE spectral estimation [GRZ_PE()]
!             will be included in interpolation loop.
!       R = Current PE range in meters.
!       REND = Range of end of skip zone if one exists.
!       RGRZ = Range at which grazing angle was computed from PE spectral estimation.
!       RMD = Range mid-way between current and last PE range.
!       RSKIP = Approximate range interval of skip zone if duct is present.
!       RSQK = Height curvature offset: current output range squared divided by
!             (2*ek*a).

subroutine graze_int( rflat, IERROR )

use apm_mod

implicit integer(kind=4) (i-n)

```

```

implicit real(kind=8) (a-h, o-z)

ierror = 0

!IPE now represents number of PE range steps for "real" run.

if( allocated( graze ) ) deallocate( graze, stat=ierror )
allocate( graze(0:ipe), stat=ierror )
if( ierror .ne. 0 ) return
graze = 0.d0
graze(0) = pi2

r = dr
rgrz = drgrz
rmd = dr2
jr1 = 1
jpe = 1
k = 0

rskip = 0.d0
per = 0.d0
rend = 0.d0

! Check for possible skip zone and set range at which PE grazing angle will
! be included.

if(( lduct ) .and. ( .not. levap ) .and. ( rflat .gt. rhor )) then
  i = 1
  do while(( grz_ray(i,2) .le. rhor ) .and. ( i .lt. igrz ))
    i = i + 1
  end do
  if( i .lt. igrz ) then
    do j = i-1, igrz-1
      rdif = grz_ray(j+1,2) - grz_ray(j,2)
      if( rdif .gt. rskip ) then
        rskip = rdif
        rend = grz_ray(j+1,2)
      end if
    end do
    if( rskip .gt. 5.d3 ) per = dmin1( rflat, rend )
  else
    per = rmax
  end if
end if

! Start interpolation.

do i = 1, ipe

! Keep all grazing angles computed from ray trace for ranges less than horizon
! range.

  if( r .le. rhor ) then
    do while(( grz_ray(k,2) .le. r ) .and. ( k .lt. igrz ))
      k = k + 1
    end do
    if( k .eq. 1 ) then
      rsqk = r**2 / 2. / aekst
      s = (hmref - yfref + antht) - rsqk
      graze(i) = dabs(datan( s / r ))
    else
      rfrac = (r - grz_ray(k-1,2)) / (grz_ray(k,2) - grz_ray(k-1,2))
      grz_ang = plint( grz_ray(k-1,1), grz_ray(k,1), rfrac )
      graze(i) = dmax1( 0.d0, grz_ang )
    end if
  else

!Get number of points from GRZ_RAY() array within PE range interval R-DR2 to R+DR2.

    icr = 0
    do j = jr1, igrz

```

```

        if(( grz_ray(j,2) .ge. rmd ) .and. ( grz_ray(j,2) .le. rmd+dr )) then
            icr = icr + 1
            jr2 = j
        end if
    end do

!Where grazing angles are available from ray trace, use these to interpolate grazing
!angle at each PE range step.

    if( icr .gt. 0 ) then
        jr1 = jr2 - icr
        if( grz_ray(jr2,2) .ge. r ) then
            do j = jr1, jr2-1
                if(( grz_ray(j,2) .le. r ) .and. ( grz_ray(j+1,2) .ge. r )) then
                    rfrac = (r - grz_ray(j,2)) / (grz_ray(j+1,2) - grz_ray(j,2))
                    graze(i) = plint( grz_ray(j,1), grz_ray(j+1,1), rfrac )
                end if
            end do
        else
            graze(i) = sum( grz_ray(jr1+1:jr2,1) ) / dble(icr)
        end if
        jr1 = jr2

    elseif(( igpe .gt. 0 ) .and. ( r .ge. per )) then

!Otherwise, use grazing angles from smooth surface PE run to for interpolation of
!grazing angles at each range step for "real" PE run.

        jpe = min0( igpe-1, idint( r / drgrz ) )
        rgrz = dble(jpe) * drgrz
        rfrac = (r - rgrz) / drgrz
        graze(i) = plint( grz_pe(jpe+1), grz_pe(jpe), rfrac )
    else
        graze(i) = 0.d0
        if( levap ) graze(i) = grz_ray(igrz,1)
    end if
end if
rmd = rmd + dr
r = r + dr

end do

if( allocated( grz_pe ) ) deallocate( grz_pe, stat=kerror )
if( kerror .ne. 0 ) ierror = kerror

if( allocated( grz_ray ) ) deallocate( grz_ray, stat=kerror )
if( kerror .ne. 0 ) ierror = kerror

end subroutine graze_int

```

A.1.16 Subroutine INTPROF

```

! ***** SUBROUTINE INTPROF *****

! Module Name: INTPROF

! Module Security Classification: UNCLASSIFIED

! Purpose: Performs a linear interpolation vertically with height on the
!          refractivity profile. Stores interpolated profile in PROFINT().

! Version Number: 1.3.0

! INPUTS:
!   Argument List: NONE
!   Common: CON, N, NLVL
!   Public: HREF(), HT(), REFREF()

! OUTPUTS:

```

```

!   Argument List: NONE
!   Common: NONE
!   Public: PROFINT()

! Modules Used: APM_MOD

! Calling Routines: APMINIT, REFINTER

! Routines Called: PLINT(function)

! GLOSSARY:

!   Input Variables: See universal glossary for common variables.
!   Output Variables: See universal glossary for common variables.

!   Local Variables:
!       HEIGHT = Height to interpolate to.
!       FRAC = Fractional height for interpolation.

```

```

SUBROUTINE intprof

```

```

  use apm_mod

```

```

  implicit integer(kind=4) (i-n)
  implicit real(kind=8) (a-h, o-z)

```

```

  j = 1
  k = 0

```

```

  profint(0) = refref(0) * con
  DO I = 1, N
    height = ht(i)
    do while( ( height .gt. href(j) ) .and. ( j .lt. nlvl ) )
      j = j + 1
      k = j - 1
    end do
    FRAC = (height - href(k)) / (href(j) - href(k))
    profint(I) = plint( refref(k), refref(j), frac ) * con
  end do

```

```

END subroutine intprof

```

A.1.17 Subroutine PEINIT

```

! ***** SUBROUTINE PEINIT *****

```

```

! Module Name: PEINIT

```

```

! Module Security Classification: UNCLASSIFIED

```

```

! Purpose: Initializes all variables used in PE model routines.

```

```

! Version Number: 1.3.1

```

```

! INPUTS:

```

```

!   Argument List: RFIX, IFLAG, IPL
!   Common: ANTH, DELZ, FK0, FK02, FTER, FREQ, N, NPROF, RMAX, RMULT, ZMAX
!   Public: SLP(), TX(), TY()
!   Data: PI, QI

```

```

! OUTPUTS:

```

```

!   Argument List: IERROR
!   Common: ACUT, CON, DELP, DR, DR2, DTHETA, DZ2, FNORM, IPE, IZINC, N34, NF4,
!           NM1, QIFK0, RHOR
!   Public: ENVPR(), FILT(), FRSP(), HT(), TYH(), U()

```

```

! Modules used: APM_MOD

```

```

! Calling Routines: APMINIT

```

```

! Routines Called:
!   APM Specific:  ALLARRAY_PE, INTPROF, XYINIT
!   Intrinsic:  ALLOCATE, ALLOCATED, CDEXP, DATAN, DBLE, DCOS, DEALLOCATE, DMAX1,
!               DMIN1, DSQRT, IDNINT

! GLOSSARY:

!   Input Variables: See universal glossary for common variables.
!   IFLAG = Integer flag indicating where in APMINIT the initializing will
!           be performed:
!           0 = called before GETGRAZE in order to compute grazing angles
!           1 = called for real PE run
!   IPL = Polarization
!           0 = horizontal
!           1 = vertical
!   RFIX = Range spacing of terrain points if equally spaced, otherwise = 0.

!   Output Variables: See universal glossary for common variables.
!   IERROR = Returning error flag:
!           0 = no error
!           non-zero = error found

!   Local Variables:
!   ADR = First coefficient in polynomial equ. for calculating DR.
!   ANG = Exponent term:
!   ANG = -i*dr*[k-sqrt(k**2-p**2)] where k is the free-space
!   wavenumber, p is the transform variable (p=k*sin(theta)), and
!   i is the imaginary number (i=sqrt(-1)).
!   B1DR = Second coefficient in polynomial eq. for calculating DR.
!   B2DR = Third coefficient in polynomial eq. for calculating DR.
!   DRT = Limiting range step value - for terrain cases.
!   RKM = Maximum range in km.

subroutine peinit( rfix, ipl, iflag, IERROR )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

complex(kind=8) ang

data adr, b1dr, b2dr / 55.67485d0, 3.52969d0, -0.01122d0 /

ierror = 0
con = 1.d-6 * fko
qifko = qi * fko
dz2 = 2.d0 * delz

! Initialize variables for free-space propagator phase calculations.

delp = pi/zmax
FNorm = 2.d0 / dble(N)
dtheta = delp / fko
nm1 = n - 1
nf4 = n / 4

! Allocate and initialize all arrays associated with PE calcs.

call allarray_pe( IERROR )

! Determine horizon range based on transmitter height and 0 receiver height.
! RHOR = SQRT[ (2*k*mean earth radius) ANTHT ]

rhor = 4121.81198 * dsqrt( antht )
acut = datan( antht / rhor )
dr = fko2 * delz**2      !Just use this as a basis, DR may change later.
rkm = rmax * 1.d-3

! Determine PE range step and integer increment at which to store

```

```

! propagation factor, angle, and range at ZLIM.
drt = adr + b1dr*rkm + b2dr*rkm**2
if( fter ) then
  dr = dmin1( dr, 700.d0 )
  dr = dmax1( dr, drt )
  if( rfix .gt. 0. ) then
    rd = rfix / dr
    if( rd .lt. 1. ) then
      dr = idnint( 1. / rd ) * rfix
    else
      dr = rfix / idnint( rd )
    end if
  end if
  izinc = 1
else
  dr = dmax1( dr, drt, 100. )
  izinc = 3
  if( freq .ge. 5000. ) izinc = 2
  if( freq .ge. 10000. ) izinc = 1
end if

!If called for initialization of GETGRAZE routine, then set lower limit on
!PE range step.

if( iflag .eq. 0 ) then
  dr = dmax1( dr, 150.d0 ) - 1.d0
else
  dr = rmult * dr
end if

dr2 = .5 * dr

!Calculate # of PE range steps.

ipe = idnint( rmax / dr ) + 1

if( fter ) then
  if( allocated( tyh ) ) deallocate( tyh, stat=ierror )
  allocate( tyh(0:ipe), stat=ierror )
  if( ierror .ne. 0 ) return
! Determine terrain elevation for each PE range.

  r = 0.
  tyh(0) = ty(1)
  slope = slp(1)
  kt = 1

  do i = 1, ipe
    r = r + dr

! Check to see if current range is past a range point in terrain profile.
! If so, increment counter, determine terrain height at current range.

    do while((r .gt. tx(kt+1)) .and. (kt .lt. itpa))
      kt = kt + 1
      slope = slp(kt)
    end do
    tyh(i) = ty(kt) + slope * ( r - tx(kt) )

  end do
end if

! Initialize variables and set-up filter array for PE calculations.

n34 = 3.* nf4
cn75 = pi / dbple(nf4)
filt = .5 + .5 * dcos( /(i, i=0, nf4)/) * cn75 )

```



```

! Define mesh array in height
do i = 0, n
    ht(i) = dble(i) * delz
end do

! Determine the free-space propagator (p-space) arrays.

DO I=0,N
    ak = dble(i) * delp
    aksq = ak * ak
    aksq = dsqrt( fko**2 - aksq )
    ang = qi*dr * ( aksq - fko )
    frsp(i) = fnorm * cdexp( ang )
end do

! Filter the upper 1/4 of the propagator arrays.

frsp(n34:n) = filt(0:nf4) * frsp(n34:n)

! If over-water and range-independent case then initialize all refractivity
! and z-space propagator arrays now.

if((.not. fter) .and. ( nprof .eq. 1 )) then
    call intprof

! Compute environmental phase term for 2nd part of PE step.

    envpr = cdexp( qi*dr * profint )

! Filter upper 1/4 of the array.

    envpr(n34:n) = filt(0:nf4) * envpr(n34:n)
end if

! Initialize starter field.

call xyinit( ipl )

end subroutine peinit

```

A.1.18 Subroutine PROFREF

```

! ***** SUBROUTINE PROFREF *****

! Module Name: PROFREF

! Module Security Classification: UNCLASSIFIED

! Purpose: This subroutine determines the refractivity profile with respect
!          to the reference height YREF which, depending on the value of IFLAG,
!          can be HMINTER or the local ground height above HMINTER.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: IFLAG, YREF
!   Common: IEXTRA, LVLEP
!   Public: HTDUM(), REFDUM()

! OUTPUTS:
!   Argument List: NONE
!   Common: LVLEP, NLVL
!   Public: HREF(), HTDUM(), REFDUM(), REFREF()

! Modules Used: APM_MOD

! Calling Routines: APMINIT, REFINIT, REFINTER

! Routines Called:

```

```

!   APM Specific: NONE
!   Intrinsic: DABS, IDINT

! GLOSSARY: See universal glossary for common variables.

!   Input Variables:
!       IFLAG = 0: Profile arrays REFREF() and HREF() will be referenced to
!                   height HMINTER, and will also be used to initialize REFDUM()
!                   and HTDUM().
!       = 1: Profile arrays REFREF() and HREF() will be referenced to the
!                   local ground height.
!       YREF = Reference height in meters at current range.

!   Output Variables:NONE

!   Local Variables:
!       FRAC = Fractional height used for interpolation
!       IBMSL = Integer flag indicating if YREF is below mean sea level (msl)
!       IBMSL = 0 -> YREF not below msl
!       IBMSL = 1 -> YREF below msl
!       JS = Integer index indicating at what index/level in array HTDUM()
!            YREF is located.
!       RMU = Interpolated M-unit value at height YREF.
!       NEWL = New/adjusted number of levels to be stored in HREF() and
!            REFREF().

subroutine profref( yref, iflag )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

nlvl = lvlep
href = 0.      !array
refref = 0.    !array

if( dabs(yref) .gt. 1.d-3 ) then

    ibmsl = 0
    js = -1

! Check to see if reference height is below mean sea level.

    if( yref .lt. 0. ) then
        ibmsl = 1
        js = 0
    end if

! Get refractivity profile level at which the height of the ground is just
! above. This level is JS.

    else
        nlvlm1 = nlvl - 1
        do i = 0, nlvlm1
            if(( yref .le. htum(i+1) ) .and. ( yref .gt. htum(i) )) js = i
        end do
    end if

! Determine the refractivity value at the ground and fill arrays HREF() and
! REFREF() with refractivity profile where height 0. now refers to the ground
! reference,i.e., either local ground height or HMINTER.

    if(( js .gt. -1 ) .or. ( ibmsl .eq. 1 )) then
        jsp1 = js + 1
        frac = (yref - htum(js)) / (htum(jsp1) - htum(js))
        rmu = refdum(js) + frac * (refdum(jsp1) - refdum(js))
        if((iextra .eq. 0) .and. (ibmsl .eq. 1)) rmu = refdum(js) + .118 * (yref-htum(js))
        if( idint( frac ) .eq. 1 ) js = jsp1
        newl = nlvl - js
        refref(0) = rmu
        href(0) = 0.
    end if
end subroutine profref

```

```

        k = js + 1
        do jk = 1, newl
            refref(jk) = refdum(k)
            href(jk) = htdum(k) - yref
            k = k + 1
        end do
        nlvl = newl
        if( iflag .eq. 0 ) then
            lvlep = nlvl
            refdum = refref      !array
            htdum = href         !array
        end if
    end if
else

! If the reference height is 0. then HREF()/REFREF() and HTDUM()/REFDUM()
! are equal.

        href = htdum
        refref = refdum
    end if

end subroutine profref

```

A.1.19 Subroutine RDTRACE

```

! *****SUBROUTINE RDTRACE*****

! Module Name: RDTRACE

! Module Security Classification: UNCLASSIFIED

! Purpose:  Performs ray trace to determine grazing angles for subsequent
!           use in rough surface calculations.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: THMXG
!   Common: ANTREF, DR, FTER, HMAX, HTLIM, IHYBRID, IPE, ISTART1, PI2, RMAX, TWOKA
!   Public: GRDUM(), HTDUM(), TYH()
!   Parameter: PI
!   Data: DEG10

! OUTPUTS:
!   Argument List: IERROR
!   Common: IGRZ
!   Public: GRZ_RAY(,)

! Modules Used: APM_MOD

! Calling Routines: GETGRAZE

! Routines Called:
!   APM Specific: NONE
!   Intrinsic: ALLOCATE, ALLOCATED, DABS, DATAN, DBLE, DEALLOCATE, DMOD, DSQRT, DTAN,
!              MAX0, MIN0, SIGN

! GLOSSARY:

!   Input Variables: See universal glossary for common and public variables.
!   THMXG = Maximum PE propagation angle for grazing angle calcs (3 degrees).

!   Output Variables: See universal glossary for common and public variables.
!   IERROR = Integer flag returning error if one occurs.

!   Local Variables:
!   A0 = Angle at start of trace in radians.
!   A1 = Angle at end of trace in radians.
!   AINC1 = Angle increment of 1e-3 degree.

```

```

!      AINC2 = Angle increment of 3e-3 degree.
!      AINC3 = Largest angle increment - approx.
!      AL = Launch angle
!      ANGMAX = Maximum angle to trace to.
!      DEGH = 1/2 degree in radians.
!      DEGT = 1.5 degrees in radians.
!      GRD = Gradient of current refractivity layer being traced through.
!      H0 = Height at start of trace in meters.
!      H1 = Height at end of trace in meters.
!      IHIT = Integer flag indicating if ray has hit surface:
!             IHIT = 0 -> has not hit surface
!             IHIT = 1 -> has hit surface
!      JL = Index for refractivity profile - current layer being traced
!            through.
!      NMAX = Maximum number of grazing angles to store in GRZ_RAY(,)
!      NR1 = Number of rays to trace at increment of AINC1.
!      NR2 = Number of rays to trace at increment of AINC2.
!      NR3 = Number of rays to trace at increment of AINC3.
!      R0 = Range at start of trace in meters.
!      R1 = Range at end of trace in meters.
!      RSQK = Height offset to account for earth curvature.
!      TSLOPE = Slope of terrain segment at current traced range.
!      YNT = Height of terrain at current traced range.

subroutine rdtrace( thmxg, IERROR )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

data nr1, nr2, nr3 / 1500, 1000, 500 / !Number of rays to trace at different
! angular intervals.
data deg1 / 1.7453292d-2 / !1 degree

! Define in line ray trace functions:

rada1( a, b ) = a**2 + 2. * grd * b !a=a0, b=h1-h0
ap( a, b ) = a + b * grd !a=a0, b=r1-r0
rp( a, b ) = a + b / grd !a=r0, b=a1-a0
hp( a, b, c ) = a + ( b**2 - c**2 ) / 2. / grd !a=h0, b=a1, c=a0

angmax = thmxg
deg1 = .5 * deg1
deg1 = deg1 * 1.5
if( ihybrid .eq. 1 ) deg1 = deg1
ainc1 = deg1 / dble(nr1) !smaller increment between +-.5 degrees
ainc2 = 2.d0 * deg1 / dble(nr2) !next larger increment between +-.5 to +-1.5 degrees
ainc3 = 2.d0 * ( thmxg - deg1 ) / dble(nr3) !largest increment

nray = nr1 + .5 * (nr2 + nr3)
nmax = 10 * nray

if( allocated( grz_ray ) ) deallocate( grz_ray, stat=ierror )
allocate( grz_ray(0:nmax,2), stat=ierror )
if( ierror .ne. 0 ) return
grz_ray = 0.
grz_ray(0,1) = pi2

ynt = 0.
if( fter ) ynt = tyh(1)

al = -angmax
igrz = 0

do nr = 1, nray

a0 = al
h0 = antref
r0 = 0.d0
jl = istart1

```

```

    iquit = 0
    icycle = 0
    tslope = 0.d0

!Special case if antenna height w.r.t. the reference height is at refractivity
!inflection point.

    if( dabs(antref-htdum(istart1)) .le. 1.d-3 ) then
        if( dabs(a0) .lt. 1.d-6 ) then
            a0 = 0.d0
            r1 = rmax
            graze_ang = 0.d0
            if( fter ) then
                k = 1
                do while(( h0 .gt. tyh(k) ) .and. (k .lt. ipe))
                    k = k + 1
                end do
                if( k .ge. ipe ) then
                    r1 = rmax
                    graze_ang = 0.d0
                else
                    hfrac = (h0 - tyh(k-1)) / (tyh(k) - tyh(k-1))
                    r1 = (dble(k-1) + hfrac) * dr
                    graze_ang = datan( (tyh(k)-tyh(k-1)) / dr )
                end if
            end if
            igrz = igrz + 1
            grz_ray(igrz,1) = graze_ang
            grz_ray(igrz,2) = r1
            al = al + ainc1
            icycle = 1
        elseif( a0 .lt. 0 ) then
            jl = jl - 1
        end if
    end if

    if( icycle .eq. 1 ) cycle
    ro = 0.d0
    ihit = 0

    do i = 1, ipe
        ro = ro + dr

        do while((r0 .lt. ro) .and. (h0 .le. htlim-h0*1.d-5) .and. (iquit .eq. 0))
            ihit = 0
            r1 = ro
            grd = grdum(jl)
            a1 = ap( a0, r1-r0 )

            if( dabs(a0) .lt. deg10 ) then

                if(( sign(1.,a0) .ne. sign(1.,a1) ) .and. (a0 .ne. 0.)) then
                    a1 = 0.
                r1 = rp( r0, a1-a0 )
                end if
                h1 = hp( h0, a1, a0 )

            else

! If more than 10 degrees, then use straight line geometry.

                rsqk = (r1-r0)**2 / twoka
                a1 = a0
                h1 = h0 + (r1 - r0) * dtan(a1) - rsqk

            end if

!After done with ray trace step, now determine current height of terrain and see if
!ray has fallen below this height.

            if( fter ) then

```

```

    tslope = (tyh(i) - tyh(i-1)) / dr
    if( dmod( r1, dr ) .gt. dr*1.d-5 ) then
        rfrac = r1 - dble(i-1)*dr
        hter = tyh(i-1) + rfrac * tslope
    else
        hter = tyh(i)
    end if

    IF( h1 .le. hter-h1*1.d-5 ) THEN
        ihit = 1

!Determine slope of line segment going into ground

        rdif = r1 - r0
        rslope = (h1 - h0) / rdif

!Determine range and height at which ray hits ground.

        r1num = tyh(i-1) - h0 - tslope*dble(i-1)*dr + rslope * r0
        r1 = r1num / ( rslope - tslope )
        h1 = h0 + rslope * ( r1 - r0 )
        if( dabs(h1) .le. 1.d-3 ) h1 = 0.d0
        if( dabs(a0) .lt. deg10 ) then
            a1 = ap( a0, r1-r0 )
        else
            rsqk = (r1-r0)**2 / twoka
            a1 = a0
            h1 = h0 + (r1 - r0) * dtan(a1) - rsqk
        end if

    END IF
end if

if(( a1 .ge. 0. ) .and. ( h1 .ge. htdum(jl+1)-1.d-5 )) then
    h1 = htdum(jl+1)
    rad = rada1( a0, h1-h0 )
    a1 = dsqrt( rad )
    r1 = rp( r0, a1-a0 )
    jl = jl + 1
elseif(( a1 .le. 0. ) .and. ( h1 .le. htdum(jl)+1.d-5 )) then
    h1 = htdum(jl)
    if( jl .eq. 0 ) ihit = 1
    rad = rada1( a0, h1-h0 )
    a1 = -dsqrt( rad )
    r1 = rp( r0, a1-a0 )
    jl = max0( 0, jl - 1 )
end if

if( h1 .ge. htlim-h1*1.d-5 ) then
    h1 = htlim
    rad = rada1( a0, h1-h0 )
    a1 = dsqrt( rad )
    r1 = rp( r0, a1-a0 )
end if

! The ray has hit surface and is reflected.

if( ihit .eq. 1 ) then
    beta = datan( tslope )
    graze_ang = beta - a1
    a1 = 2. * beta - a1
    igrz = min0( igrz + 1, nmax )
    grz_ray(igrz,1) = graze_ang
    grz_ray(igrz,2) = r1
    if( dabs(grz_ray(igrz,2) - grz_ray(igrz-1,2)) .le. 1.d-3 ) then
        igrz = igrz - 1
        h0 = hmax+1.d0
        iquit = 1
    end if
end if

```

```

        h0 = h1
        r0 = r1
        a0 = a1
        if( a0 .ge. pi2 ) iquit = 1
    end do
end do

if( dabs(al) .lt. deg ) then
    al = al + ainc1
elseif( dabs(al) .lt. degt ) then
    al = al + ainc2
else
    al = al + ainc3
end if

end do

! Sort temporary grazing angle array.

k = 1
do while( k .gt. 0 )
    k = 0
    do j = 1, igrz-1
        if( grz_ray(j,2) .gt. grz_ray(j+1,2) ) then
            k = j
            hk = grz_ray(j,2)
            grz_ray(j,2) = grz_ray(j+1,2)
            grz_ray(j+1,2) = hk
            hk = grz_ray(j,1)
            grz_ray(j,1) = grz_ray(j+1,1)
            grz_ray(j+1,1) = hk
        end if
    end do
end do

end subroutine rdtrace

```

A.1.20 Subroutine REFINIT

```

! ***** SUBROUTINE REFINIT *****

! Module Name: REFINIT

! Module Security Classification: UNCLASSIFIED

! Purpose: Initializes refractivity arrays used for subsequent PE and RO
!          calculations.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: NONE
!   Common: ANTREF, FTER, HMINTER, IHYBRID, LERR12, LVLP,
!           NPROF, RMAX, TROP0, YFREF
!   Public: HMSL(,), REFMSL(,), RNGPROF(), TY()

! OUTPUTS:
!   Argument List: HTRAP, HTHICK, IERROR, RMMIN, RMMAX
!   Common: ACRIT, IS, ISTART, ISTART1, LDUCT, LEVAP, LEVELS, LVLEP, LVLP, NLVL,
!           RMTX, RV2, SNREF_TX
!   Public: HMSL(,), HTDUM(,), GR(,), GRDUM(,), Q(,), REFDUM(,), REFMSL(,),
!           RM(,), ZRT()

! Modules Used: APM_MOD

! Calling Routines: APMINIT

! Routines Called:
!   APM Specific: PROFREF, REMDUP
!   Intrinsic: ANY, DABS, DMAX1, DSQRT, IDINT, MAXVAL, MINLOC

```

```

! GLOSSARY: See universal glossary for common variables

!   Input Variables: NONE

!   Output Variables:

!       HTHICK = Thickness in meters of highest trapping layer from all
!               refractivity profiles.
!       HTRAP = Height of highest trapping layer in meters from all
!               refractivity profiles.
!       IERROR = -12 : Range of last refractivity profile entered (for
!               range dependent case) is less than RMAX. (This is
!               returned from subroutine REFINIT). Will only
!               return this error if error flag LERR12 is set to
!               .TRUE.).
!               = -13 : Height of first level in any user-specified refrac-
!               tivity profile is greater than 0. First height must
!               be at m.s.l. (0.) or >0. if below m.s.l.
!               = -14 : Last gradient in user-provided refractivity profile
!               is negative.
!       RMMAX = Maximum M-unit value (x10e-6) of refractivity profile at or
!               below antenna height at range 0.
!       RMMIN = Minimum M-unit value (x10e-6) of refractivity profile at
!               range 0.

!   Local Variables:

!       GRD = Gradient at current height/refractivity level.
!       HLRGE = Maximum height limit for last level in height/refractivity
!               profiles.
!       HTE = Height of minimum M-unit of 1st refractivity profile.
!       IHTE = Index level of HTE.
!       RCRIT = Minimum M-unit value (x10e-6) of refractivity profile above
!               transmitter height at range 0.
!       ZHI = Height at next higher profile point in meters.
!       ZLO = Height at next lower profile point in meters.

subroutine refinit( HTRAP, HTHICK, RMMIN, RMMAX, IERROR )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

dimension ihtel(1), icrh(1)

data hlarge/ 1.d6 /

ierror = 0
rmmax = 0.
rmmmin = 0.
rmtx = 0.
levap = .false.
lduct = .false.

! Test to see if last profile entered ( for range dependent case ) meets or
! exceeds RMAX, otherwise, return error (unless error trapping is turned off -
! LERR12 = .FALSE.).

if((nprof .gt. 1) .and. ((rngprof(nprof) .lt. rmax) .and. (lerr12))) ierror = -12

! Test to see that every user-specified profile begins with 0. height for
! 1st level in profile (allow for <0. height if below m.s.l.).

if( any(hmsl(0,:) .gt. 0.d0 )) ierror = -13
if( ierror .ne. 0 ) return

do i = 1,nprof

! Test on HDIF greater than 0 for profiles that contain multiple height/M-unit values

```


! that are equal. LVLP is already one more than # of actual levels in profiles.

```

    hdif = 0.
    lvlm1 = lvlp
    lvlm2 = lvlp
    do while( hdif .le. 1.d-6 )      !Ignore multiple height/ref. pairs
        lvlm1 = lvlm1 - 1
        lvlm2 = lvlm1 - 1
        hdif = hmsl(lvlm1,i) - hmsl(lvlm2,i)
    end do
    grd = (refmsl(lvlm1,i)-refmsl(lvlm2,i)) / hdif

! If last gradient in refractivity profile is negative then return error.

    if( grd .lt. 0 ) then
        ierror = -14
        return
    end if

! Add extra level to tabulated profiles with extrapolated gradient.

    hmsl(lvlp, i) = hlarge
    refmsl(lvlp, i) = (hlarge-hmsl(lvlm1,i)) * grd + refmsl( lvlm1, i )
end do

is = 1
rv2=rngprof(is)

refdum(0:lvlp) = refmsl( 0:lvlp, is )
htdum(0:lvlp) = hmsl( 0:lvlp, is )

lvlep = lvlp

! Remove any duplicate levels in first profile and adjust HTDUM() and
! REFNUM() to minimum terrain height.

call remdup
call profref( hminter, 0 )

! Determine surface refractivity at source range 0. for troposcatter calcs.

snref_tx = 0.d0
if( tropo ) then
    call profref( yfref, 1 )
    snref_tx = refref(0)
end if

! NLVL is now the number of height/refractivity levels in adjusted HTDUM()
! and REFNUM(). Find height and thickness of highest trapping layer, if one
! exists, relative to HMINTER.

htrap = 0.
hthick = 0.
do i = 1, nprof
    do j = 0, lvlp-1
        grd = refmsl(j+1,i) - refmsl(j,i)
        hp1 = hmsl(j+1,i) - hminter
        if(( grd .lt. 0. ) .and. (hp1 .gt. htrap)) then
            htrap = hp1
            hp0 = hmsl(j,i) - hminter
            hthick = hp1 - hp0
        end if
    end do
end do

! Build gradient array and determine index level of antenna height.

istart1 = 0
do i = 0, lvlep-1
    if( antref .ge. htdum(i) ) istart1 = i
    refdif = refdum(i+1)-refdum(i)

```

```

        if( dabs( refdif ) .lt. 1.d-2 ) refdum(i+1) = refdum(i) + 1.d-2
        grdum(i) = 1.d-6 * (refdum(i+1)-refdum(i)) / (htdum(i+1)-htdum(i))
    end do

    ! Build Z and RM arrays for R0 calculations if needed. Add level for
    ! ANTH, if needed.

    if( ihybrid .eq. 1 ) then

        zrt(0) = htdum(0)
        rm(0) = 1.d-6 * refdum(0)
        i = 0
        istart = 0

        DO j = 1, lvlep
            zhi = htdum(j)
            zlo = htdum(j-1)
            i = i + 1
            IF (dABS(zhi - antref) .LT. 1.d-3) istart = i
            IF ((istart .EQ. 0) .AND. (zhi .GT. antref)) THEN
                zrt(i) = antref
                ip1 = i + 1
                im1 = i - 1
                zrt(ip1) = zhi
                rm(ip1) = 1.d-6 * refdum(j)
                if( dabs( rm(ip1) - rm(im1) ) .lt. 1.d-8 ) rm(ip1) = rm(im1) + 1.d-8
                drmdz = (rm(ip1) - rm(im1)) / (zrt(ip1) - zrt(im1))
                rm(i) = rm(im1) + drmdz * (antref - zrt(im1))
                istart = i
                i = i + 1
            ELSE
                zrt(i) = zhi
                rm(i) = 1.d-6 * refdum(j)
                if( dabs( rm(i) - rm(i-1) ) .lt. 1.d-8 ) rm(i) = rm(i-1) + 1.d-8
            END IF
        END DO

        ! Highest profile point exceeds antenna height. Total number of
        ! points in z array reduced by 1 since highest level is not needed.

        levels = i - 1

        ! For special case when ground is initially flat, but at non-zero
        ! height, re-adjust all R0 refractivity arrays.

        if(( dabs(ty(1)) .gt. 1.d-3 ) .and. (fter)) then
            nl = levels
            yref = ty(1)
            href = 0.
            refref = 0.
            js = -1

            ! Get refractivity profile level at which the height of the ground is just
            ! above. This level is JS.

            do i = 0, nl
                if(( yref .le. zrt(i+1) ) .and. ( yref .gt. zrt(i) ))      js = i
            end do

            ! Determine the refractivity value at the ground and fill arrays HREF() and
            ! REFREF() with refractivity profile where height 0. now refers to the ground
            ! reference, i.e., either local ground height or HMINTER.

            if( js .gt. -1 ) then
                jsp1 = js + 1
                frac = (yref - zrt(js))/(zrt(jsp1) - zrt(js))
                rmu = rm(js) + frac * (rm(jsp1) - rm(js))
                if( idint( frac ) .eq. 1 ) js = jsp1
                newl = nl - js
                refref(0) = rmu
                href(0) = 0.
            end if
        end if
    end if

```

```

        k = js + 1
        do jk = 1, newl
            refref(jk) = rm(k)
            href(jk) = zrt(k) - yref
            k = k + 1
        end do
        levels = newl
        rm(:levels) = refref(:levels)
        zrt(:levels) = href(:levels)
    end if

! Re-determine antenna height index.

        istart = istart - js

    end if

! Build GR and Q arrays for ray-optics and ray-tracing routines.

        do i = 0, levels
            ip1 = i + 1
            rmd = rm(ip1) - rm(i)
            gr(i) = rmd / (zrt(ip1) - zrt(i))
            q(i) = 2. * rmd
        end do

        rmtx = rm(istart)

else

        rmtx = refdum(istart1) + grdum(istart1) * 1.d6 * (antref - htdum(istart1))
        rmtx = rmtx * 1.d-6

end if

! Determine minimum (1.E-6*M) on profile above transmitter height.

rcrit = 0.d0
ist = istart1 + 1
icrh = minloc( refdum(ist:) ) + ist
icr = icrh(1) - 1
rcrit = refdum(icr) * 1.d-6

! Determine minimum (1.E-6*M) on profile, and maximum M at or below
! the transmitter.

ihtel = minloc( refdum )
ihthe = ihtel(1) - 1      ! Subtract 1 because function MINLOC returns
rmmin = refdum(ihthe)    ! element starting with "1".
hte = htdum(ihthe)
rmmax = dmax1( rmtx*1.d6, maxval( refdum(0:istart1) ) )

rmmin = rmmin * 1.d-6
rmmax = rmmax * 1.d-6

! Compute critical angle if antenna is within duct.

acrit = 0.
if( rcrit .le. rmtx ) acrit = dsqrt( 2.d0*(rmtx - rcrit) ) + 1.d-6
if( ihthe .gt. 0 ) lduct = .true.

! Test to see if evaporation duct profile has been specified.

if(( nprof .eq. 1 ) .and. ( htrap .gt. 0. )) then
    if(( hte .le. 40. ) .and. ( hte .gt. 0. )) then
        gr2 = (grdum(ihthe) - grdum(ihthe-1)) * 1.d6
        if( dabs( gr2 ) .le. .1 ) levap = .true.
    end if
end if

end subroutine refinit

```

A.1.21 Subroutine REMDUP

```
! ***** SUBROUTINE REMDUP *****
! Module Name: REMDUP
! Module Security Classification: UNCLASSIFIED
! Purpose: Removes duplicate refractivity levels in profile.
! Version Number: 1.3.0
! INPUTS:
!   Argument List: NONE
!   Common: LVLEP
!   Public: HTDUM(), REFDUM()
! OUTPUTS:
!   Argument List: NONE
!   Common: LVLEP
!   Public: HTDUM(), REFDUM()
! Modules Used: APM_MOD
! Calling Routines: APMINIT, REFINIT, REFINTER
! Routines Called:
!   APM Specific: NONE
!   Intrinsic: DABS
! GLOSSARY:
!   Input Variables: See universal glossary for common variables.
!   Output Variables: See universal glossary for common variables.
subroutine remdup
use apm_mod
implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)
! Remove all duplicate levels in interpolated profile
i = 0
do while( i .lt. lvlep )
  ht1 = htdum(i)
  ht2 = htdum(i+1)
  if( dabs(ht1-ht2) .le. 1.d-3 ) then
    lvlep = lvlep - 1
    do j = i, lvlep
      jp1 = j + 1
      htdum(j) = htdum(jp1)
      refdum(j) = refdum(jp1)
    end do
    i = i - 1
  end if
  i = i + 1
end do
end subroutine remdup
```

A.1.22 Subroutine TERINIT

```
! ***** SUBROUTINE TERINIT *****
! Module Name: TERINIT
```

```

! Module Security Classification: UNCLASSIFIED

! Purpose: This routine initializes the arrays TX() and TY() and all
!          associated terrain variables.

! Version Number: 1.3.1

! INPUTS:
!   Argument List: NONE
!   Common: ANTH, HMAX, HMIN, ITP, ITPA, LERR6, RMAX, RUF
!   Public: DIELEC(), IGRND(), TERX(), TERY()

! OUTPUTS:
!   Argument List: ANGU, HTERMAX, IERROR, RFIX, RFLAT
!   Common: ANTREF, FTER, HMINTER, HMREF, HTLIM, RUF
!   Public: SLP(), TX(), TY()

! Modules Used: APM_MOD

! Calling Routines: APMINIT

! Routines Called:
!   APM Specific: NONE
!   Intrinsic: ALL, ANY, DABS, DATAN, DBLE, DMAX1, IDNINT, MAXVAL, MINVAL

! GLOSSARY: See universal glossary for common variables and parameters.

!   Input Variables: NONE

!   Output Variables:
!     ANGU = Maximum tangent ray angle from source to terrain peak
!           along profile path.
!     HTERMAX = Maximum terrain height along profile path in meters.
!     IERROR = Integer value that is returned if any errors exist in input
!             data:
!             -6 : Last range in terrain profile is less than RMAX.
!                   (Will only return this error if error flag LERR6
!                   is set to .TRUE.).
!             -8 : HMAX is less than maximum height of terrain profile.
!             -9 : Antenna height w.r.t. msl is greater than maximum
!                   height HMAX.
!             -17 : Range points of terrain profile are not increasing.
!             -18 : First range point is not 0.
!     RFIX = If terrain profile points are equally spaced, this is
!            automatically determined and range spacing is set to RFIX,
!            otherwise, RFIX = 0.
!     RFLAT = Maximum range in meters at which the terrain profile
!            remains flat from the source.

!   Local Variables:
!     ANGLE = Tangent angle from source to each terrain point in radians
!     HDEG = 1/2 degree in radians.
!     RDIF1 = Difference between adjacent terrain point elevations.
!     RDIF2 = Difference between next adjacent terrain point elevations.
!     RDIFSUM = Running sum of adjacent terrain point differences.
!     RFRAC = Maximum fraction between adjacent terrain point differences.
!     SLOPE = Slope of terrain segment.
!     SLPTOL = Terrain slope "fudge" factor.
!     X1, X2 = Range of Ith and I+1 terrain point, respectively.
!     Y1, Y2 = Height of Ith and I+1 terrain point, respectively.
!     XDIF = Range difference between adjacent terrain points.
!     YDIF = Height difference between adjacent terrain points.

subroutine terinit( ANGU, RFIX, RFLAT, HTERMAX, IERROR )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

```

```

data slptol / 1.d-5 /      ! tolerance of terrain slope
data hdeg / 8.726646d-3 /  ! 1/2 degree

fter = .false.
ierror = 0
angu = 0.
hminter = 0.
antref = antht
htermax = 0.
rflat = rmax

if( itp .gt. 0 ) fter = .true.

! Check that all terrain range points are increasing.

if( fter ) then
  do i = 1, itp-1
    if( terx(i+1) .lt. terx(i) ) then
      ierror = -17
      return
    end if
  end do

! Test to see that first range value is 0.

  if( terx(1) .gt. 0.d0 ) then
    ierror = -18
    return
  end if

! Test to see if the last range point in the profile meets or exceeds RMAX.
! If not then return error code.

  if( terx(itp) .lt. rmax ) then
    if( lerr6 ) then
      ierror = -6
      return
    end if
  end if

! Determine if terrain profile points are spaced at fixed increments.

  rdif1 = terx(2)-terx(1)
  rfrac = 0.
  rdifsum = rdif1
  do i = 2, itp-1
    rdif2 = dmax1( 1.d-3, terx(i+1) - terx(i) )
    rdifff = rdif2 / rdif1
    if( rdifff .gt. rfrac ) rfrac = rdifff
    rdifsum = rdifsum + rdif2
    rdif1 = rdif2
  end do

! If it is determined that terrain points are spaced at fixed range
! increments, then set this increment = RFIX.

  rfix = 0.
  if( rfrac .lt. 1.05 ) rfix = idnint( rdifsum / dble(itp-1) )

! Determine minimum height of terrain profile.

  hminter = minval( tery )

! Then adjust entire terrain profile by this minimum height HMINTER
! such that this is the new 0 reference. Get maximum height of terrain,
! store adjusted terrain profile in arrays TX() and TY().

  ty(1:itp) = tery(1:itp) - hminter
  tx(1:itp) = terx(1:itp)

! Return error code if HMAX does not exceed the maximum height of the

```

```

! terrain profile.

htermmax = maxval( tery )
if( htermmax .gt. hmax ) then
    ierror = -8
    return
end if

! Add extra point to working terrain profile arrays TX() and TY().

if( tx(itp) .lt. rmax ) then
    tx(itpa) = rmax * 1.1
else
    tx(itpa) = tx(itp) * 1.1
end if
ty(itpa) = ty(itp)

antref = antht + ty(1)

iswitch = 0
do i = 1, itpa-1

    y1 = ty(i)
    x1 = tx(i)
    ip1 = i + 1
    y2 = ty(ip1)
    x2 = tx(ip1)

    xdif = x2 - x1
    ydif = y2 - y1
    xdif = dmax1( xdif, 1.d-5 )
    slope = ydif / xdif

    slp(i) = slope

! Determine first range point at which terrain profile is no longer flat.

    if( ( dabs(slope) .gt. slptol ) .and. ( iswitch .eq. 0 ) ) then
        rflat = tx(i)
        iswitch = 1
    end if

! Calculate angle made from each terrain point height to antenna height above
! reference (HMINTER). Determine maximum propagation angle so that direct ray
! angle will clear highest peak.

    if( y1 .gt. antref ) then
        angle = datan( (y1-antref) / x1 ) !angle from reflected ray
        if( angle .gt. angu ) angu = angle
    end if

end do

! Add 1/2 degree to the angle that clears the highest peak.

angu = angu + hdeg

! Test if entire terrain profile is over land. If it is, and rough surface
! calculations have been specified, then turn RUF flag off.

if( ruf ) then
    if( ( all(igrnd .gt. 1) ) .and. ( all(igrnd .lt. 7) ) ) then      !All terrain
        ruf = .false.
    elseif( any(igrnd .eq. 7) ) then      !May be sea or fresh water
        if( any(dielec(1,:) .lt. 30. ) ) ruf = .false.      !Just test on relative
permittivity
    end if
    end if
    end if
    !value for sea or fresh water.

end if
end if

```

```

hmref = hmin - hminter
htlim = hmax - hminter

! Return error if antenna height is greater than maximum plot height.
if( antref .ge. htlim-1.d-3 ) ierror = -9

end subroutine terinit

```

A.1.23 Subroutine TRACE_ROUT

```

!***** SUBROUTINE TRACE_ROUT *****

! Module Name: TRACE_ROUT

! Module Security Classification: UNCLASSIFIED

! Purpose: This routine traces a single ray to each output range ROUT
!          and stores the height reached in array HARRAY.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: AS, GRDUM(), HS, HTDUM(), HTLIM, JLS, LVLEP, NROUT,
!                  RNGOUT(), RS
!   Common: NONE
!   Public: NONE

! OUTPUTS:
!   Argument List: HARRAY(), IHMX
!   Common: NONE
!   Public: NONE

! Modules Used: NONE

! Calling Routines: FILLHT, GETTHMAX

! Routines Called:
!   APM Specific: NONE
!   Intrinsic: DABS, DMIN1, DSIGN, DSQRT, MIN0

! GLOSSARY:

!   Input Variables:
!     AS = Initial launch angle in radians
!     GRDUM() = Array of refractivity gradients defined by profile HTDUM(),
!              REFDUM()
!     HS = Initial height in meters
!     HTDUM() = Height array containing height values for current (interpolated)
!              profile in meters, relative to HMINTER.
!     HTLIM = user-supplied maximum height relative to HMINTER, i.e.,
!              HTLIM = HMAX - HMINTER
!     JLS = Starting refractivity profile level
!     LVLEP = Number of height/refractivity levels in profile REFDUM(),HTDUM()
!              taken w.r.t. reference height HMINTER.
!     NROUT = integer number of output range points desired
!     RNGOUT() = array containing all output ranges in meters.
!     RS = Initial range in meters

!   Output Variables:
!     HARRAY() = Height of ray at each output range.
!     IHMX = Output range step index where maximum height HTLIM is reached in
!            array HARRAY().

!   Local Variables:
!     A0 = Angle at start of trace in radians.
!     A1 = Angle at end of trace in radians.
!     GRD = Gradient of current refractivity layer being traced through.
!     H0 = Height at start of trace in meters.
!     H1 = Height at end of trace in meters.

```



```

!      IQUIT = Integer flag indicating when to terminate trace loop.
!      JL = Index for refractivity profile - current layer being traced
!           through.
!      JR = Integer index counter.
!      R0 = Range at start of trace in meters.
!      R0 = Current range to trace to.
!      R1 = Range at end of trace in meters.

subroutine trace_rout( as, rs, hs, jls, nrout, htlim, grdum, htdum, &
                     lvlep, rngout, IHMX, HARRAY )

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

real(kind=8) grdum(0:*), htdum(0:*), rngout(*), harray(*)

! Define in line ray trace functions:

rada1( a, b ) = a**2 + 2. * grd * b           !a=a0, b=h1-h0
rp( a, b ) = a + b / grd                     !a=r0, b=a1-a0
ap( a, b ) = a + b * grd                     !a=a0, b=r1-r0
hp( a, b, c ) = a + ( b**2 - c**2 ) / 2. / grd !a=h0, b=a1, c=a0

ihmx = 0
a0 = as
r0 = rs
h0 = hs
jl = jls
jr = 1

do while( r0 .ge. rngout(jr) )
    jr = jr + 1
end do

harray(1:jr) = 0.
iquit = 0

do while(( iquit .eq. 0 ) .and. ( jr .le. nrout ))
    harray(jr) = 0.
    ro = rngout(jr)

    do while(( r0 .lt. ro ) .and. ( h0 .le. htlim-h0*1.d-5 ))

        r1 = ro
        grd = grdum(jl)
        a1 = ap( a0, r1-r0 )
        if( dsign(1.,a0) .ne. dsign(1.,a1) ) then
            a1 = 0.
            r1 = rp( r0, a1-a0 )
        end if
        h1 = hp( h0, a1, a0 )

        if( h1 .gt. (htdum(jl+1) - h1*1.d-5) ) then
            h1 = dmin1( htlim, htdum(jl+1) )
            rad = rada1( a0, h1-h0 )
            a1 = dsqrt( rad )
            r1 = rp( r0, a1-a0 )
            jl = min0( lvlep, jl + 1 )
        elseif( h1 .gt. htlim - h1*1.d-5 ) then
            h1 = htlim
            rad = rada1( a0, h1-h0 )
            a1 = dsqrt( rad )
            r1 = rp( r0, a1-a0 )
            iquit = 1
        end if

        a0 = a1
        h0 = h1
        r0 = r1

    end do

end do

```

```

        harray(jr) = h0
        if( dabs(h0-htlim) .le. 1.d-5 ) iquit = 1
        jr = jr + 1
    end do

    if( jr .le. nrout ) then
        harray(jr:nrout) = htlim
        ihmz = jr
    end if

end subroutine trace_rout

```

A.1.24 Subroutine TROPOINT

```

!***** SUBROUTINE TROPOINT *****

! Module Name: TROPOINT

! Module Security Classification: UNCLASSIFIED

! Purpose: This routine initializes all variables and arrays need for
!          troposcatter loss computations.

! Version Number 1.3.0

! INPUTS:
!   Argument List: NONE
!   Common: ANTREF, FREQ, FTER, IPE, NROUT, NZOUT
!   Public: RNGOUT(), TYH(), ZOUT()

! OUTPUTS:
!   Argument List: NONE
!   Common: AEK, EK, JT2, THETA1S, TLSTWR, TWOKA, R1T, RF,
!   Public: ADIF(), D2S(), RDT(), TH1(), THETA0(), THETA2S()

! Modules Included: APM_MOD

! Calling Routines: APMINIT

! Routines Called:
!   APM Specific: ANTPAT, GET_K
!   Intrinsic: ALLOCATE, ALLOCATED, DBLE, DEALLOCATE, DLOG10, DSQRT

! GLOSSARY:

!   Input Variables: See universal glossary for common variables.

!   Output Variables: See universal glossary for common variables.

!   Local Variables:
!       ALD = Log of antenna pattern factor for ALPHAD where ALPHAD here
!             represents lowest direct ray angle in optical region.
!       D1 = Range of each terrain point in meters
!       D1S = Tangent range in meters for source height over smooth
!             surface.
!       D2 = Tangent range in meters for output receiver height Z
!             over smooth surface.
!       FACTR = Antenna pattern factor for angle ALPHAD.
!       H1 = Height of each terrain point in meters
!       QA = Term for determining horizon range.
!       RDHOR1 = Minimum range (in meters) at which diffraction field
!                solutions are applicable and intermediate region ends,
!                for smooth surface and 0 receiver height.

subroutine tropoint
use apm_mod

```

```

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

if( fter ) then
  if( allocated( th1 ) ) deallocate( th1, stat=ierror )
  allocate( th1(ipe), stat=ierror )
  if( ierror .ne. 0 ) return
  th1 = 0.
end if

!Obtain effective earth radius factor.

call get_k( 1 )

!Initialize THETA0 angle array.

theta0 = rngout / aek

!Initialize terms used in calculation of troposcatter loss.

rf = .0419 * freq
rit = rf * antref

!Initialize range to tangent point, D1S, and tangent angle,
!THETA1S, for source over smooth surface.

ald = 0.
d1s = dsqrt( twoka * antref )
theta1s = d1s / aek

!Get antenna pattern loss term, ALD, based on smooth earth tangent
!angle.

alphad = theta1s + 1.d-6
call antpat( alphad, FACTR )
if( factr .ne. 0. ) ald = 20. * dlog10( factr )

!Determine the minimum range, RDHOR1, at which diffraction field
!solutions are applicable and intermediate region ends, for smooth
!surface and 0 receiver height.

qa = 230200.d0 * ( ek**2 / freq )**0.3333333
rdhor1 = dsqrt( twoka * antref ) + qa      !in meters

!Initialize tangent range and tangent angle, D2S & THETA2S, (for smooth
!surface) for all output receiver heights.

do i = 0, nzout
  z = zout(i)
  if( z .lt. 0. ) cycle
  d2 = dsqrt( twoka * z )
  theta2s(i) = d2 / aek
  d2s(i) = d2

!Determine minimum range, RDT(), at which diffraction field
!solutions are applicable and intermediate region ends (for smooth
!surface) for all output receiver heights. Initialize ADIF() for use
!in TROP0 routine.

  rdt(i) = rdhor1 + d2
  adif(i) = antref - z
end do

!For terrain, determine all tangent angles, TH1().

if( fter ) then
  ald = 0.
  do i = 1, ipe
    h1 = tyh(i)
    d1 = dble(i) * dr
    ang1 = (antref - h1) / d1 + d1 / twoka
  end do
end if

```

```

        th1(i) = ang1
    end do

!Initialize array index counter.

    jt2 = 1
end if

!Initialize troposcatter loss term.

tlstwr = 54.9 + 30. * dlog10( freq ) - ald

end subroutine tropoint

```

A.1.25 Subroutine XYINIT

```

! ***** SUBROUTINE XYINIT *****

! Module Name: XYINIT

! Module Security Classification: UNCLASSIFIED

! Purpose:  Determines the initial PE starter field.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: IPOLAR
!   Common: ANTH, DTHETA, FK0, LN, N, N34, WL, ZMAX
!   Public: FILT()

! OUTPUTS:
!   Argument List: NONE
!   Common: ALPHAD
!   Public: U()

! Modules Used: APM_MOD

! Called Routines: PEINIT

! Routines Called:
!   APM Specific: ANTPAT, DRST
!   Intrinsic: CMPLX, DASIN, DBLE, DCONJG, DCOS, DIMAG, DSIN, DSQRT, REAL

! GLOSSARY: See universal glossary for common variables.

!   Input Variables:
!       IPOLAR = Antenna polarization:
!           0 = Horizontal
!           1 = Vertical

!   Output Variables: NONE

!   Local Variables:
!       ANTKO = Free space wavenumber * antenna height
!       ATTN = Attenuation factor.
!       DTERM = Exponential phase term for real source.
!       FACD = Antenna pattern factor for direct angle
!       FACR = Antenna pattern factor for reflected (image) angle
!       PK = Sine of angle.
!       RTERM = Exponential phase term for image source.
!       SGAIN = Normalization factor.
!       ZPK = Phase term for real and image sources.

SUBROUTINE xyinit( ipolar )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

```

```

complex(kind=8) rterm, dterm

sgain= dsqrt( wl ) / zmax
antko = fko * antht

! Construct initial field in p-space.

DO I=0,N

    pk = dble(i) * dtheta
    aksq = pk * pk
    cak = (1. - aksq)**(-.75)
    zpk = pk * antko

! Get antenna pattern factors for the direct and reflected rays.

    alphad = dasin( pk )
    call antpat( alphad, FACD )
    call antpat( -alphad, FACR )

    rterm = cmplx( dcos( zpk ), dsin( zpk ), 8 )
    dterm = dconjg( rterm )

    if( ipolar .eq. 0 ) u(i) = cak*sgain * ( facd * dterm - facr * rterm ) !H pol
    if( ipolar .eq. 1 ) u(i) = cak*sgain * ( facd * dterm + facr * rterm ) !V pol

end do

! Filter upper 1/4 of the field.

u(n34:n) = filt(0:nf4) * u(n34:n)

! Transform to z-space.

itsw = 1 - ipolar

udum(0:n) = real( u(0:n), 8 )
call drst( udum, ln, itsw )
if( ipolar .eq. 0 ) udum = -udum
u(0:n) = cmplx( udum(0:n), dimag(u(0:n)), 8 )

udum(0:n) = dimag( u(0:n) )
call drst( udum, ln, itsw )
if( ipolar .eq. 0 ) udum = -udum
u(0:n) = cmplx( real( u(0:n), 8 ), udum(0:n), 8 )

END subroutine xyinit

```

A.2 SUBROUTINE APMSTEP

```

! ***** SUBROUTINE APMSTEP *****

! Module Name: APMSTEP

! Module Security Classification: UNCLASSIFIED

! Purpose: This routine advances and computes the propagation loss for
!          one output range.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: ISTEP
!   Common: GASATT, HTLIM, IHYBRID, IO, IZG, NOPE, NROUT, NZOUT, PEFLAG,
!           RATZ, RPEST
!   Public: HTFE(), RNGOUT(), ZOUT()
!   Data: RTST

! OUTPUTS:

```

```

!   Argument List: JEND, JSTART, MPFL(), ROUT
!   Common: GASLOSS, JT2

!   Modules Used: APM_MOD

!   Calling Routines: MAIN DRIVER PROGRAM

!   Routines called:
!   APM Specific: AIRBORNE, FEM, PESTEP, ROLOSS
!   Intrinsic: MAX0, IIDNNT

!   GLOSSARY: See universal glossary for common variables and parameters.

!   Input Variables:
!       ISTP = Current output range step index.

!   Output Variables:
!       JEND = index at which the valid propagation loss values end.
!       JSTART = index at which the valid propagation loss values begin.
!       ROUT = current output range in meters.
!       MPFL(,) = 2-byte integer array containing propagation factor(F) and loss
!                 values in centibels vs. height, at each output range ROUT.
!                 All values returned are referenced to height HMIN.
!                 MPFL(1,i) = loss at output height i*DZOUT
!                 MPFL(2,i) = 20*log10(F) at output height i*DZOUT

!   Local Variables:
!       JAE = ending index within MPFL() of FE(airborne) loss values.
!       JAS = starting index within MPFL() of FE(airborne) loss values.
!       JFE = ending index within MPFL() of FE loss values.
!       JFS = starting index within MPFL() of FE loss values.
!       JPE = ending index within MPFL() of PE loss values.
!       JPS = starting index within MPFL() of PE loss values.
!       JRE = ending index within MPFL() of RO loss values.
!       JRS = starting index within MPFL() of RO loss values.

subroutine apmstep( istp, ROUT, MPFL, JSTART, JEND )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

integer(kind=2) mpfl(2,0:*)

rout = rngout(istp)

!Compute loss in dB due to gaseous absorption. GASATT is 0 if appropriate parameters
!have not been specified.

gasloss = rout * gasatt

jps = 0
jpe = 0
jrs = 0
jre = 0
jfs = 0
jfe = 0
jas = 0
jae = 0

!Advance and compute the field for one output range step.

if( peflag ) then
    call pestep( istp, rout, mpfl, JPS, JPE )
    jstart = jps
else
    if( nope .eq. 0 ) call pestep( istp, rout, mpfl, JPS, JPE )

```

```

jstart = jps

if( ihybrid .eq. 0 ) then      !airborne hybrid model
! Compute loss for lower FE region.

    jas = max0( 0, izg ) + io
    j = 0
    do while( htfe(istp) .gt. zout(j) )
        j = j + 1
    end do
    jae = j - 1
    if( jas .lt. jae ) call airborne( rout, MPFL, jas, jae, istp, 1 )
    jstart = jas

! Compute loss for upper FE region.

    jas = jpe+1
    jae = nzout
    if( jpe .lt. jae ) call airborne( rout, MPFL, jas, jae, istp, 0 )

elseif(( ihybrid .eq. 1 ) .and. ( rout .lt. ratz )) then !full hybrid

    if( rout .lt. rtst ) then
        jfs = max0( 0, izg ) + io
        jfe = nzout
        jstart = jfs
    else
        if( htfe(istp) .lt. (htlim - htfe(istp)*1.d-5) ) then
            j = nzout
            do while( zout(j) .gt. htfe(istp) )
                j = j - 1
            end do
            jfs = max0( jpe+1, j+1 )
            jfe = nzout
        end if
    end if

    if( jfe .gt. 0 ) call fem( istp, rout, MPFL, jfs, jfe )

! Get loss based on RO calculations from ZOUT(JRS) to ZOUT(JRE).

    if( rout .ge. rtst ) then
        jre = jfs - 1
        if( jfe .eq. 0 ) jre = nzout
        jrs = jpe + 1
        if( rout .lt. rpest ) jrs = max0( 0, izg ) + io

        if( jrs .gt. jre ) then
            jrs = 0
            jre = 0
        end if
    end if

    if( jre .gt. 0 ) call roloss( istp, rout, jrs, jre, MPFL )
end if

end if

jend = max0( jfe, jre, jpe, jae )

! Reset counter for calling TROPO from XOSTEP routine.

if( istp .eq. nrout ) jt2 = 1

end subroutine apmstep

```

A.2.1 Subroutine AIRBORNE

```

! ***** SUBROUTINE AIRBORNE *****

```

```

! Module Name: AIRBORNE

! Module Security Classification: UNCLASSIFIED

! Purpose: This routine determines propagation loss based on flat
!          earth calculations for all output heights specified at each
!          output range ROUT.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: IFLAG, ISTP, JAE, JAS, ROUT
!   Common: GASATT, PLCNST, TERANG, TWOKA, TWOKA_DOWN
!   Public: RSQRD(), ZOUTMA()

! OUTPUTS:
!   Argument List: MPFL()
!   Common: NONE

! Modules Used: APM_MOD

! Calling Routines: APMSTEP

! Routines called:
!   APM Specific: ANTPAT
!   Intrinsic: DATAN, DLOG10, DMAX1, DSQRT, IIDNNT

! GLOSSARY: See universal glossary for common variables.

!   Input Variables:
!       IFLAG = Integer flag indicating if calculations are to be per-
!               formed for heights above upper angular PE region (IFLAG=0)
!               or below angular PE region (IFLAG=1)
!       JAE = ending index within MPFL() of FE(airborne) loss values.
!       JAS = starting index within MPFL() of FE(airborne) loss values.
!       ROUT = Current output range in meters.

!   Output Variables:
!       MPFL(,) = 2-byte integer array containing propagation factor(F) and loss
!               values in centibels vs. height, at each output range ROUT.
!               All values returned are referenced to height HMIN.
!               MPFL(1,i) = loss at output height i*DZOUT
!               MPFL(2,i) = 20*log10(F) at output height i*DZOUT

!   Local Variables:
!       ALPHAX = Direct ray angle.
!       DLOSS = Propagation loss in dB.
!       FACD = Antenna pattern factor for direct ray.
!       FFACDB = Pattern propagation factor in dB
!       FSLOSS = Free-space in dB.
!       R1 = Path length of direct ray
!       RSQ = Square of output range ROUT
!       RSQK = Height curvature offset: current output range squared divided by
!               (2*ek*a).
!       ZAB = Height of desired output point relative to real source height
!               with earth curvature offset.

subroutine airborne( rout, MPFL, jas, jae, istp, iflag )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

integer(kind=2) mpfl(2,0:*)

rsq = rsqrd(istp)
if( iflag .eq. 0 ) rsqk = rsq / twoka           !above PE region
if( iflag .eq. 1 ) rsqk = rsq / twoka_down      !below PE region

```



```

! Begin loop for calculations of FE loss for heights from ZOUT(JAS) to
! ZOUT(JAE).

do i = jas, jae

  zab = zoutma(i) - rsqk
  alphax = datan( zab / rout )

  if(( alphax .gt. terang ) .or. ( iflag .eq. 0 )) then

    call antpat( alphax, FACD )
    r1 = dsqrt( zab**2 + rsq )

    ffacdb = 20. * dlog10( dmax1( facd, 1.d-13 ) )

    fsloss = 20. * dlog10( r1 ) + plcns

!Compute loss in dB due to refractive and absorption effects.

    dloss = fsloss - ffacdb + r1 * gasatt
    ffacdb = fsloss - dloss

    mpfl(1,i) = iidnnt( 10. * dloss )      !convert real*8 number to int*2
    mpfl(2,i) = iidnnt( 10. * ffacdb )    !convert real*8 number to int*2

  end if

end do

end subroutine airborne

```

A.2.2 Subroutine CALCLOS

```

! ***** SUBROUTINE CALCLOS *****

! Module Name: CALCLOS

! Module Security Classification: UNCLASSIFIED

! Purpose: Determines the PE propagation loss at each output range step
!          ROUT and all heights up to ZLIM.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: ISTD, RLAST
!   Common: ANTREF, DELZ, DR, DZOUT, FTER, GASLOSS, HMREF, IHMX, IHYBRID, IO, IXO,
!           NZOUT, PEFLAG, RATZ, RLOG, RLOGLST, RPEST, TROPO, TWOKA_DOWN, YCUR,
!           YLAST, ZLIM
!   Public: FSL(), HLIM(), HTFE(), RNGOUT(), U(), ULST(), ZOUT()
!   Data: TERRAIN, INVAL

! OUTPUTS:
!   Argument List: JEND, JSTART, MPFL()
!   Common: IZG, TERANG
!   Public: FFROUT(), RFAC1(), RFAC2(), RLOSS()

! Modules Used: APM_MOD

! Calling Routines: PESTEP

! Routines called:
!   APM Specific: GETPFAC(function), PLINT(function), TROPOSCAT
!   Intrinsic: MAX0, MIN0, DATAN, DMAX1, DMIN1, IDINT, IDNINT, IIDNNT

! GLOSSARY: See universal glossary for common variables, public variables,
!            and parameters.

!   Input Variables:
!     ISTD = index of output range step

```

```

!      RLAST = last PE range in meters

!      Output Variables:
!      JEND = index at which valid loss values in MPFL() ends.
!      JSTART = index at which valid loss values in MPFL() begin.
!      MPFL(,) = 2-byte integer array containing propagation factor(F) and loss
!               values in centibels vs. height, at each output range ROUT.
!               All values returned are referenced to height HMIN.
!               MPFL(1,i) = loss at output height i*DZOUT
!               MPFL(2,i) = 20*log10(F) at output height i*DZOUT

!      Local Variables:
!      AG = Tangent angle from antenna height to terrain elevation at
!           current range ROUT.
!      FF = Propagation factor in dB at range ROUT and specified height.
!      IP1 = Index in array RFAC1() corresponding to ground height at
!            previous PE range. All array elements in RFAC1() from 1 to
!            IP1 are set equal to PFACMIN.
!      IP2 = Index in array RFAC2() corresponding to ground height at
!            current PE range. All array elements in RFAC2() from 1 to
!            IP2 are set equal to PFACMIN.
!      ROUT = Current output range in meters.
!      XX = Fractional range at which to interpolate propagation factor
!           for current output range ROUT.
!      YCH = Height of terrain at current PE range relative to
!            reference height HMREF.
!      YLH = Height of terrain at previous PE range relative to
!            reference height HMREF.
!      ZDIF = Height from antenna to terrain elevation at current range
!            ROUT, adjusted for earth curvature.
!      ZEND2 = Height at which to stop calculating propagation factor.
!      ZHT = Height at which to compute propagation factor.
!      ZINT = Interpolated ground height at current output range ROUT.

subroutine calclos( rlast, istp, MPFL, JSTART, JEND )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

integer(kind=2) mpfl(2,0:*)

rout = rngout(istp)
ych = ycur - hmref
ylh = ylast - hmref

! Get height of ground at output range ROUT and determine number of vertical
! output points that correspond to the ground height. Fill the loss array
! MPFL() with INTERRAIN (-999) to represent ground for those vertical output
! points.

xx = (rout - rlast) / dr
zint = plint( ylast, ycur, xx )
izg = idint( (zint-hmref) / dzout )
mpfl(:,0:izg) = interrains

jstart = max0( 0, izg ) + io

! Determine current maximum angle for lower FE region if using airborne
! hybrid model (IHYBRID=0).

if( ihybrid .eq. 0 ) then
    zdif = zint - antref - rout**2/twoka_down
    ag = datan( zdif / rout )
    terang = dmax1( ag, terang )
end if

! If current output range is greater than RPEST then begin calculation of
! loss values and return them in MPFL().

```

```

if( rout .ge. rpest ) then

! Determine values of array elements corresponding to the ground and set these
! to the minimum propagation factor (-300) for later interpolation.

ip1 = 0
ip2 = 0

if( fter ) then

    ip1 = idint( ylh / dzout )
    ip2 = idint( ych / dzout )
    ip1 = max0( 0, ip1 )
    ip2 = max0( 0, ip2 )

    if( zout(ip1)-ylast .lt. 0. ) ip1 = ip1 + 1
    if( zout(ip2)-ycur .lt. 0. ) ip2 = ip2 + 1

    rfac1(0:ip1-1) = 0.
    rfac2(0:ip2-1) = 0.

    if(( jstart .lt. ip1 ) .and. ( jstart .lt. ip2 )) then
        it = min0(ip1, ip2)
        izg = max0( izg, it )
        mpfl(:,izg) = interrain
        jstart = max0( 0, izg ) + io
    end if

end if

! Determine height/integer value at which to stop calculating loss.
! NOTE: For terrain cases, ray tracing was performed
! using the direct ray angle and sometimes HLIM(i) may be less than the
! local ground height. The GOTO statement is used just as a safety factor
! in this case.

if( peflag ) then
    jend = max0( 0, idint( (zlim-hmref) / dzout ) )
else
    zend1 = dmax1( zint, hlim(istp) )
    zend2 = dmin1( zlim, zend1 )
    if(( istp .ge. ihmz ) .or. ( rout .ge. ratz )) then
        jend = max0( 0, idnint( (zend2-hmref) / dzout ) )
    else
        jend = max0( 0, idint( (zend2-hmref) / dzout ) )
    end if
end if
jend = min0( jend, nzout )

if( jend .lt. jstart ) goto 5

! Get propagation factor at valid heights from field at previous PE range.

if( rloglst .gt. 0. ) then
    do i = ip1, jend
        zht = zout(i) - ylast
        rfac1(i) = getpfac( ulst, rloglst, delz, zht )
    end do
end if

! Get propagation factor at valid heights from field at current PE range.

do i = ip2, jend
    zht = zout(i) - ycur
    rfac2(i) = getpfac( u, rlog, delz, zht )
end do

! If using PE & XO model, determine what heights in MPFL()
! will contain invalid loss and set equal to INVALID(-1000).

if( ihybrid .eq. 2 ) then

```

```

        jstart1 = idint( (htfe(istp) - hmref) / dzout )
        mpfl(:,jstart:jstart1) = inval
        jstart = max0( jstart, jstart1+io )
    end if

! If using full hybrid model or PE & XO model, determine the
! propagation factor at the last point in height in the PE region. This
! is used for subsequent interpolation in the XO model.

    if( ixo .gt. 0 ) then
        z1 = zlim - ylast
        z2 = zlim - ycur
        rf1 = getpfac( ulst, rloglst, delz, z1 )
        rf2 = getpfac( u, rlog, delz, z2 )
        ff = plint( rf1, rf2, xx )
        ffrout(istp,1) = ff
        ffrout(istp,2) = zlim-zint
    end if

! Interpolate between the current and last PE range to get propagation loss
! at range ROUT. Compute troposcatter loss for total loss contribution.
! Note: RFAC1,2() arrays contain 20*log10(F).

    do k = jstart, jend
        if( rloglst .gt. 0. ) then
            ff = plint( rfac1(k), rfac2(k), xx )
            if(( k .ge. ip1 ) .and. ( k .lt. ip2 )) ff = rfac1(k)
            if(( k .lt. ip1 ) .and. ( k .ge. ip2 )) ff = rfac2(k)
            rloss(k) = fsl(istp) - ff
        else
            rloss(k) = fsl(istp) - rfac2(k)
        end if
    end do

! Compute troposcatter loss.

    if( tropo ) call troposcat( istp, jstart, jend )

! Include gaseous absorption loss.

    rloss(jstart:jend) = rloss(jstart:jend) + gasloss

!Convert real*8 number to int*2 and store propagation loss and propagation factor.

    mpfl(1,jstart:jend) = iidnnt( 10.* rloss(jstart:jend) )
    mpfl(2,jstart:jend) = iidnnt( 10.* ( fsl(istp) - rloss(jstart:jend) ) )

5 continue

! Fill remainder of array with -1000 indicating non-valid loss values.

    jn = jend + 1
    mpfl(:,jn:nzout) = inval

else

! If current output range is less than RPEST then there are no current valid
! loss values at any height - fill MPFL() with -1000. JSTART and JEND will be
! equal and will have a value of 0 or 1 (depending on polarization) if smooth
! surface case, otherwise will have a value of the nearest integer multiple of
! DZOUT corresponding to the height of the local ground.

    mpfl(:,jstart:nzout) = inval
    jend = jstart

end if

end subroutine calclos

```

A.2.3 Subroutine DOSHIFT

```

! ***** SUBROUTINE DOSHIFT *****
!
! Module Name: DOSHIFT
!
! Module Security Classification: UNCLASSIFIED
!
! Purpose: Shifts the PE field by the # of bins corresponding to height of
!          the ground.
!
! Version Number: 1.3.0
!
! INPUTS:
!   Argument List: NONE
!   Common: DELZ, N, NM1, YCUR, YLAST
!   Public: U()
!
! OUTPUTS:
!   Argument List: NONE
!   Public: U()
!
! Modules Used: APM_MOD
!
! Calling Routines: GETGRAZE, PESTEP
!
! Routines called:
!   APM Specific: NONE
!   Intrinsic: DABS, IDNINT
!
! GLOSSARY:
!
!   Input Variables:
!     See universal glossary for common variables
!
!   Output Variables:
!     See universal glossary for common variables
!
!   Local Variables:
!     INCR = Integer indicating which direction to shift field U().
!           INCR = 1 -> terrain slope is positive, shift down.
!           INCR = -1 -> terrain slope is negative, shift up.
!     KBIN = Integer # of bins or mesh heights to shift.
!     YDIF = Height difference between current and last ground elevation.
!
subroutine doshift
  use apm_mod

  implicit integer(kind=4) (i-n)
  implicit real(kind=8) (a-h, o-z)

  ! Determine # of bins to shift field.

  ydif = ycur - ylast
  kbin = idnint( dabs(ydif) / delz )
  if( kbin .eq. 0 ) return

  ! If slope is positive then shift array elements down.

  if( ydif .ge. 0. ) then
    incr = 1
    jst = 1
    jend = nm1 - kbin
  else

  ! If slope is negative then shift array elements up.

    incr = -1
    jst = nm1
    jend = kbin + 1
  end if

```

```

kinc = incr * kbin
do j = jst, jend, incr
  jk = j + kinc
  u(j) = u(jk)
end do

if( incr .gt. 0 ) then
  nst = n - kbin
  u(nst:nml) = 0.
else
  u(1:kbin) = 0.
end if

end subroutine doshift

```

A.2.4 Subroutine DRST

```

SUBROUTINE DRST(X,N,IFLAG)
!
! *****
!
!Obtained from:
! Dan Dockery
! APL - Johns Hopkins Univ.
! Baltimore, MD

!With minor code modifications by:
! Amalia Barrios
! SPAWARSSYSCEN D858
! San Diego, CA

!   DRST REPLACES THE REAL*8 VECTOR X BY ITS FINITE DISCRETE SINE OR C
!   TRANSFORM.  THE ALGORITHM IS BASED ON A MIXED RADIX (8-4-2)
!   REAL VECTOR FAST FOURIER SYNTHESIS ROUTINE PUBLISHED BY
!   (G. D. BERGLAND, "A RADIX-EIGHT FAST FOURIER TRANSFORM SUBROUTINE
!   FOR REAL-VALUED SERIES," IEEE TRANSACTIONS ON AUDIO AND ELECTRO-
!   ACOUSTICS, VOL. AU-17, PP. 138-144, JUNE 1969) AND SINE AND COSINE
!   TRANSFORM ALGORITHMS PUBLISHED BY COOLEY, LEWIS, AND WELSH
!   (J. W. COOLEY, P. A. W. LEWIS AND P. D. WELSH, "THE FAST FOURIER
!   TRANSFORM ALGORITHM PROGRAMMING CONSIDERATIONS IN THE CALCULATION
!   OF SINE, COSINE AND LAPLACE TRANSFORMS," J. EM VIB., VOL. 12,
!   PP. 315-337, JULY 1970).

!   NOTE:  CALLS FOR THE SINE TRANSFORM RETURN THE DISCRETE ANALOG
!           OF -1 TIMES THE SINE TRANSFORM INTEGRAL

!           MODIFIED JULY 1983 BY J. P. SKURA

! -----
!           INPUT PARAMETERS
!
!           X      A 2**N+1 REAL*8 ARRAY FOR THE TRANSFORM
!
!           N      TRANSFORM SIZE
!
!           IFLAG   FLAG TO SIGNIFY WHICH TRANSFORM TO PERFORM
!                   IFLAG=0 FOR COSINE TRANSFORM
!                   IFLAG=1 FOR SINE TRANSFORM
!                   iflag = -1 deallocates all allocated arrays
! -----
!
!           OUTPUT PARAMETERS
!
!           X      A 2**N+1 POINT REAL*8 ARRAY OF THE TRANSFORMED DATA
!
!           N      TRANSFORM SIZE (UNCHANGED)

```

```

!      IFLAG      UNCHANGED

!-----

!      TABLES - ARRAY      REQUIRED DIMENSIONS
!      B          2**N + 1
!      ST         2**N
!      JI         2**(N-1) - 1
!      CS         2**(N-4) - 1
!      SS         2**(N-4) - 1
!
!      SUBROUTINES - DR8SYN (RADIX 8 SYNTHESIS)
!                   DR4SYN (RADIX 4 SYNTHESIS)
!                   DR2TR  (RADIX 2 TRANSFORM)

!*****

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

DIMENSION X(*)

double precision, allocatable :: b(:), st(:), cs(:), ss(:)
integer(kind=4), allocatable :: ji(:)

save n2, n4, n8, np, npd2, npd4, npd16, npm1, nmax2, nmax16
save b, st, cs, ss, ji

DATA N2 / 0 /

IF(( N.NE.N2 ) .and. ( iflag .ge. 0 )) then

    N2=N
    NP=2**N2
    nmax2 = np / 2
    nmax16 = np / 16

    if( allocated ( b ) ) deallocate ( b )
    allocate ( b(np+1) )
    b = 0.

    if( allocated ( ji ) ) deallocate ( ji )
    allocate ( ji(nmax2) )
    ji = 0

    if( allocated ( st ) ) deallocate ( st )
    allocate ( st(np) )
    st = 0.

    if( allocated ( cs ) ) deallocate ( cs )
    allocate ( cs(nmax16) )
    cs = 0.

    if( allocated ( ss ) ) deallocate ( ss )
    allocate ( ss(nmax16) )
    ss = 0.

!      COMPUTE CONSTANTS AND CONSTRUCT TABLES

    N8=N2/3
    N4=N2-3*N8-1
    NPD2=NP/2
    NPD4=NP/4
    NPD16=NP/16
    NPM1=NP-1
    DT=PI/dbl(NP)

    ST(1)=0.0
    DO J=1,NPM1

```

```

        T=DT*db1e(J)
        ST(J+1)=0.5d0/DSIN(T)
    end do

!    CONSTRUCT THE BIT REVERSED SUBSCRIPT TABLE.

    J1=0

    NT=NPD2-1
    DO J=1,NT
        J2=NPD2
        do while (IAND(J1,J2).NE.0)
            J1=IABS(J1-J2)
            J2=J2/2
        end do
        J1=J1+J2
        JI(J)=J1
    end do

    IF (N8.NE.0) then

!    CONSTRUCT THE TRIGONOMETRIC TABLES FOR THE RADIX 8 PASSES.
!    THE TABLES ARE STORED IN BIT REVERSED ORDER.

        J1=0
        NT=NPD16-1
        DO J=1,NT
            J2=NPD16
            do while (IAND(J1,J2).NE.0)
                J1=IABS(J1-J2)
                J2=J2/2
            end do
            J1=J1+J2
            T=DT*db1e(J1)
            CS(J)=DCOS(T)
            SS(J)=-DSIN(T)
        end do
    end if

elseif( iflag .eq. -1 ) then

!End of APM run - deallocate arrays and return to main driver program.

        if( allocated( b ) ) deallocate( b, stat = ierror )
        if( allocated( st ) ) deallocate( st, stat = ierror )
        if( allocated( cs ) ) deallocate( cs, stat = ierror )
        if( allocated( ss ) ) deallocate( ss, stat = ierror )
        if( allocated( ji ) ) deallocate( ji, stat = ierror )
        n2 = 0
        return

    end if

    IF (IFLAG.GT.0) then

!    SET UP ARRAY FOR THE SINE TRANSFORM

        B(1)=-(X(2)+X(2))
        B(2)=X(NP)+X(NP)

        J1=0
        DO J=3,NPM1,2
            J1=J1+1
            J2=JI(J1)
            J3=NP-J2
            B(J)=X(J2)-X(J2+2)
            B(J+1)=X(J3+1)
        end do

    else

```



```

!      SET UP THE ARRAY FOR THE COSINE TRANSFORM

B(1)=X(1)
B(2)=X(NP+1)
J1=0
XSUM=X(2)
DO J=3,NPM1,2
    J1=J1+1
    J2=JI(J1)
    J3=NP-J2
    XSUM=XSUM+X(J+1)
    B(J)=X(J2+1)
    B(J+1)=X(J3+2)-X(J3)
end do

end if

!      BEGIN FAST FOURIER SYNTHESIS

IF (N8.ne.0) then

!      RADIX 8 ITERATIONS

IQNT=1
NT=NPD16
DO J=1,N8
    J1= 1+IQNT
    J2=J1+IQNT
    J3=J2+IQNT
    J4=J3+IQNT
    J5=J4+IQNT
    J6=J5+IQNT
    J7=J6+IQNT
    CALL DR8SYN(IQNT, nt, cs, ss, B,B(J1),B(J2),B(J3),B(J4),B(J5),B(J6),B(J7))
    NT=NT/8
    IQNT=8*IQNT
end do
end if

if( n4 .gt. 0 ) then

!      RADIX 4 ITERATION

IQNT=NPD4
J1= 1+IQNT
J2=J1+IQNT
J3=J2+IQNT
CALL DR4SYN(IQNT,B,B(J1),B(J2),B(J3))

elseif( n4 .eq. 0 ) then

!      RADIX 2 ITERATION

IQNT=NPD2
J1= 1+IQNT
CALL DR2TR(IQNT,B,B(J1))

end if

J1=NP

IF (IFLAG.GT.0) then

!      FORM SINE TRANSFORM

DO J=2,NPD2
    S=B(J1)-B(J)
    T=B(J)+B(J1)
    X(J)=0.25*(T*ST(J)+S)
    X(J1)=0.25*(T*ST(J1)-S)
    J1=J1-1

```

```

        end do
        IQNT=NPD2+1
        X(IQNT)=0.25*(ST(IQNT)*(B(IQNT)+B(IQNT)))
        X(1)=0.0
        X(NP+1)=0.0

    else

        !      FORM THE COSINE TRANSFORM

        X(1)=.5d0*B(1)+XSUM
        X(NP+1)=.5d0*B(1)-XSUM
        B(1)=B(1)+XSUM
        B(NP+1)=B(1)-XSUM
        DO J=2,NPD2
            S=B(J1)-B(J)
            T=B(J)+B(J1)
            X(J)=.25d0*(S*ST(J)+T)
            X(J1)=.25d0*(T-S*ST(J1))
            J1=J1-1
        end do

        IQNT=NPD2+1
        X(IQNT)=0.25d0*(B(IQNT)+B(IQNT))

    end if

    RETURN
    END subroutine drst

SUBROUTINE DR8SYN(IQNT, nt, cs, ss, B0,B1,B2,B3,B4,B5,B6,B7)
! *****
!
!      RADIX 8 SYNTHESIS SUBROUTINE
!      CALLED BY DRST, THE SINE TRANSFORM DRIVER.
!
! *****

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

DIMENSION B0(*),B1(*),B2(*),B3(*),B4(*),B5(*),B6(*),B7(*)
dimension cs(*), ss(*)

DATA R2,CPI4/1.41421356237309505d0,0.7071067811865476d0/
DATA CPI8,SPI8/0.9238795325112868d0,0.3826834323650898d0/

        JT=0
        JL=2
        JR=2
        JI=3
        INT8=8*IQNT

        DO K=1,IQNT
            T0=B0(K)+B1(K)
            T1=B0(K)-B1(K)
            T2=B2(K)+B2(K)
            T3=B3(K)+B3(K)
            T4=B4(K)+B6(K)
            T5=B4(K)-B6(K)
            T6=B7(K)-B5(K)
            T7=B7(K)+B5(K)
            T8=R2*(T7-T5)
            T5=R2*(T7+T5)
            TT0=T0+T2
            T2=T0-T2
            TT1=T1+T3
            T3=T1-T3
            T4=T4+T4
            T6=T6+T6
            B0(K)=TT0+T4

```

```

      B4(K)=TT0-T4
      B1(K)=TT1+T5
      B5(K)=TT1-T5
      B2(K)=T2+T6
      B6(K)=T2-T6
      B3(K)=T3+T8
      B7(K)=T3-T8
end do

      IF (NT.EQ.0) GO TO 70

      K0=INT8+1
      KLAST=K0+IQNT-1

DO K=K0, KLAST
      T1=B0(K)+B6(K)
      T3=B0(K)-B6(K)
      T2=B7(K)-B1(K)
      T4=B7(K)+B1(K)
      T5=B2(K)+B4(K)
      T7=B2(K)-B4(K)
      T6=B5(K)-B3(K)
      T8=B5(K)+B3(K)
      B0(K)=(T1+T5)+(T1+T5)
      B4(K)=(T2+T6)+(T2+T6)
      T5=T1-T5
      T6=T2-T6
      B2(K)=R2*(T6+T5)
      B6(K)=R2*(T6-T5)
      T1=T3*CPI8+T4*SPI8
      T2=T4*CPI8-T3*SPI8
      T3=T8*CPI8-T7*SPI8
      T4=-T7*CPI8-T8*SPI8
      B1(K)=(T1+T3)+(T1+T3)
      B5(K)=(T2+T4)+(T2+T4)
      T3=T1-T3
      T4=T2-T4
      B3(K)=R2*(T4+T3)
      B7(K)=R2*(T4-T3)
end do

      GO TO 70

76      C1=CS(JT)
      S1=SS(JT)
      C2=C1*C1-S1*S1
      S2=C1*S1+C1*S1
      C3=C1*C2-S1*S2
      S3=C2*S1+S2*C1
      C4=C2*C2-S2*S2
      S4=C2*S2+C2*S2
      C5=C2*C3-S2*S3
      S5=C3*S2+S3*C2
      C6=C3*C3-S3*S3
      S6=C3*S3+C3*S3
      C7=C3*C4-S3*S4
      S7=C4*S3+S4*C3

      K=JI*INT8
      J0=JR*INT8+1
      JLAST=J0+IQNT-1

DO J=J0, JLAST
      K=K+1
      TR0=B0(J)+B6(K)
      TR1=B0(J)-B6(K)
      TI0=B7(K)-B1(J)
      TI1=B7(K)+B1(J)
      TR2=B4(K)+B2(J)
      TI3=B4(K)-B2(J)
      TI2=B5(K)-B3(J)

```

```

      TR3=B5(K)+B3(J)
      TR4=B4(J)+B2(K)
      T0=B4(J)-B2(K)
      TI4=B3(K)-B5(J)
      T1=B3(K)+B5(J)
      TR5=CPI4*(T1+T0)
      TI5=CPI4*(T1-T0)
      TR6=B6(J)+B0(K)
      T0=B6(J)-B0(K)
      TI6=B1(K)-B7(J)
      T1=B1(K)+B7(J)
      TR7=-CPI4*(T0-T1)
      TI7=-CPI4*(T0+T1)
      T0=TR0+TR2
      TR2=TR0-TR2
      T1=TI0+TI2
      TI2=TI0-TI2
      T2=TR1+TR3
      TR3=TR1-TR3
      T3=TI1+TI3
      TI3=TI1-TI3
      T5 =TI4+TI6
      TTR6=TI4-TI6
      TI6=TR6-TR4
      T4 =TR4+TR6
      T7 =TI5+TI7
      TTR7=TI5-TI7
      TI7=TR7-TR5
      T6 =TR5+TR7
      B0(J)=T0+T4
      B0(K)=T1+T5
      B1(J)=C1*(T2+T6)-S1*(T3+T7)
      B1(K)=C1*(T3+T7)+S1*(T2+T6)
      B2(J)=C2*(TR2+TTR6)-S2*(TI2+TI6)
      B2(K)=C2*(TI2+TI6)+S2*(TR2+TTR6)
      B3(J)=C3*(TR3+TTR7)-S3*(TI3+TI7)
      B3(K)=C3*(TI3+TI7)+S3*(TR3+TTR7)
      B4(J)=C4*(T0-T4)-S4*(T1-T5)
      B4(K)=C4*(T1-T5)+S4*(T0-T4)
      B5(J)=C5*(T2-T6)-S5*(T3-T7)
      B5(K)=C5*(T3-T7)+S5*(T2-T6)
      B6(J)=C6*(TR2-TTR6)-S6*(TI2-TI6)
      B6(K)=C6*(TI2-TI6)+S6*(TR2-TTR6)
      B7(J)=C7*(TR3-TTR7)-S7*(TI3-TI7)
      B7(K)=C7*(TI3-TI7)+S7*(TR3-TTR7)
end do

      JR=JR+2
      JI=JI-2
      IF (JI.GT.JL) GO TO 70
      JI=JR+JR-1
      JL=JR
70    JT=JT+1
      IF (JT.LT.NT) GO TO 76

RETURN
END subroutine DR8SYN

SUBROUTINE DR4SYN(IQNT,B0,B1,B2,B3)
! *****
!
!   RADIX 4 SYNTHESIS SUBROUTINE
!   CALLED BY DRST, THE SINE TRANSFORM DRIVER.
!
! *****

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

DIMENSION B0(*),B1(*),B2(*),B3(*)

```

```

      DO K=1,IQNT
        T0=B0(K)+B1(K)
        T1=B0(K)-B1(K)
        T2=B2(K)+B2(K)
        T3=B3(K)+B3(K)
        B0(K)=T0+T2
        B2(K)=T0-T2
        B1(K)=T1+T3
        B3(K)=T1-T3
      end do

RETURN
END subroutine dr4syn

SUBROUTINE DR2TR(IQNT,B0,B1)
! *****
!
!   RADIX 2 TRANSFORM SUBROUTINE
!   CALLED BY DRST, THE SINE TRANSFORM DRIVER.
!
! *****

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

DIMENSION B0(*),B1(*)

      DO K=1,IQNT
        T=B0(K)+B1(K)
        B1(K)=B0(K)-B1(K)
        B0(K)=T
      end do

RETURN
END subroutine dr2tr

```

A.2.5 Subroutine FEM

```

! ***** SUBROUTINE FEM *****

! Module Name: FEM

! Module Security Classification: UNCLASSIFIED

! Purpose: This routine determines propagation loss based on flat
!          earth calculations for all output heights specified at each
!          output range ROUT.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: ISTD, JFE, JFS, ROUT
!   Common: ANTREF, FKO, GASATT, HTLIM, PLCNST, TWOKA
!   Public: RSQRD(), ZOUTMA(), ZOUTPA()

! OUTPUTS:
!   Argument List: MPFL()
!   Common: ALPHAD

! Calling Routines: APMSTEP, XOSTEP

! Modules Used: APM_MOD

! Routines called:
!   APM Specific: ANTPAT, GETREFCOEF
!   Intrinsic: DATAN, DCOS, DLOG10, DMAX1, DSQRT, IIDNNT

! GLOSSARY: See universal glossary for common variables.

!   Input Variables:

```

```

!      ROUT = Current output range in meters.
!      ISTEP = Index of current output range step.

!      Output Variables:
!      JFE = Ending index within MPFL() of FE loss values.
!      JFS = Starting index within MPFL() of FE loss values.
!      MPFL(,) = 2-byte integer array containing propagation factor(F) and loss
!               values in centibels vs. height, at each output range ROUT.
!               All values returned are referenced to height HMIN.
!               MPFL(1,i) = loss at output height i*DZOUT
!               MPFL(2,i) = 20*log10(F) at output height i*DZOUT

!      Local Variables:
!      ALPHAR = Reflected ray angle.
!      DLOSS = Propagation loss in dB.
!      FACD = Antenna pattern factor for direct ray.
!      FACR = Antenna pattern factor for reflected ray.
!      FFAC2 = Square of pattern propagation factor.
!      FFACDB = Pattern propagation factor squared in dB
!      PHDIF = Total phase difference between direct and reflected ray,
!              including phase change upon reflection.
!      R1 = Path length of direct ray
!      R2 = Path length of reflected ray
!      REFCOEF = Complex reflection coefficient.
!      RMAG = Magnitude of reflection coefficient.
!      RPHASE = Phase of reflection coefficient.
!      RSQ = Square of output range ROUT.
!      RSQK = Current output range squared divided by TWOKA (2*ek*a).
!      XREFLECT = Range at which ray is reflected.
!      ZM = Height of desired output point relative to real source height
!           with earth curvature offset.
!      ZP = Height of desired output point relative to imaginary source
!           height (for reflected ray) with earth curvature offset.

subroutine fem( istp, rout, MPFL, jfs, jfe )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

integer(kind=2) mpfl(2,0:*)
complex(kind=8) refcoef

rsq = rsqrd(istp)
rsqk = rsq / twoka

xreflect = 0.d0

! Begin loop for calculations of FE loss for heights from ZOUT(JFS) to
! ZOUT(JFE).

do i = jfs, jfe

    if( rout .le. rtst ) then
        zm = zoutma(i)
        zp = zoutpa(i)
    else
        zm = zoutma(i) - rsqk
        zp = zoutpa(i) - rsqk
    end if

!Determine point of reflection.

    xreflect = rout * antref / zp

! ALPHAD = direct ray angle
! ALPHAR = reflected ray angle (grazing angle = -ALPHAR)

    alphad = datan( zm / rout )
    alphar = datan( zp / rout )

```

```

    call antpat( alphad, FACD )
    call antpat( -alphar, FACR )

    r1 = dsqrt( zm*zm + rsq )
    r2 = dsqrt( zp*zp + rsq )

! Determine reflection coefficient.

    call getrefcoef( 0, alphar, xreflect, REFCOEF, RMAG, RPHASE )

! Now get total phase lag and compute propagation factor and loss.

    phdif = ( r2 - r1 ) * fko + rphase
    frterm = facr * rmag
    ffac2 = facd*facd + frterm*frterm + 2. * facd * frterm * dcos(phdif)

    ffacdb = 10. * dlog10( dmax1( ffac2, 1.d-25 ) )

    fsloss = 20. * dlog10( r1 ) + plcnst
    dloss = fsloss - ffacdb + r1 * gasatt !Include gaseous absorption loss.
    ffacdb = fsloss - dloss

    mpfl(1,i) = iidnnt( 10. * dloss )      !convert real*8 number to int*2
    mpfl(2,i) = iidnnt( 10. * ffacdb )     !convert real*8 number to int*2

end do

end subroutine fem

```

A.2.6 Subroutine FFT

```

! ***** SUBROUTINE FFT *****

! Module Name: FFT

! Module Security Classification: UNCLASSIFIED

! Purpose: Performs fast Fourier sine transform on complex array U.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: UXY()
!   Common: LN, N

! OUTPUTS:
!   Argument List: UXY()
!   Common: NONE
!   Public: UDUM()

! Modules Used: APM_MOD

! Calling Routines: FRSTP

! Routines called:
!   APM Specific: DRST
!   Intrinsic: CMPLX, DIMAG, REAL

! GLOSSARY: See universal glossary for common variables and parameters.

!   Input Variables:
!       UXY() = Complex field to be transformed.

!   Output Variables:
!       UXY() = Transform of complex field.

subroutine fft( UXY )

use apm_mod

```

```

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

complex(kind=8) uxy(0:*)

udum(0:n) = real( uxy(0:n), 8 )
call drst( udum, ln, isn )
uxy(0:n) = cmplx( -udum(0:n), dimag(uxy(0:n)), 8 )

udum(0:n) = dimag( uxy(0:n) )
call drst( udum, ln, isn )
uxy(0:n) = cmplx( real( uxy(0:n), 8 ), -udum(0:n), 8 )

end subroutine fft

```

A.2.7 Subroutine FRSTP

```

! ***** SUBROUTINE FRSTP *****

! Module Name: FRSTP

! Module Security Classification: UNCLASSIFIED

! Purpose: Propagates the field FARRAY() in free space by one range step.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: FARRAY()
!   Common: NM1
!   Public: FRSP()

! OUTPUTS:
!   Argument List: FARRAY()
!   Common: NONE

! Calling Routines: GETGRAZE, MIXEDFT, PESTEP

! Routines called:
!   APM Specific: FFT
!   Intrinsic: NONE

! GLOSSARY: See universal glossary for common variables.

!   Input Variables:
!       FARRAY() = Field array to be propagated one range step in free
!                   space (z-space).

!   Output Variables:
!       FARRAY() = Free-space propagated field (returned in z-space).

!   Local Variables: NONE

subroutine frstp( FARRAY )
use apm_mod
complex(kind=8) farray(0:*)

call fft( FARRAY )          !Transform to Fourier space

!Multiply by free-space propagator
farray(0:nm1) = farray(0:nm1) * frsp(0:nm1)

call fft( FARRAY )          !Transform back to z-space

end subroutine frstp

```


A.2.8 Subroutine FZLIM

```
! ***** SUBROUTINE FZLIM *****
!
! Module Name: FZLIM
!
! Module Security Classification: UNCLASSIFIED
!
! Purpose: This routine stores the range, propagation factor in dB,
!          and determines and stores the outgoing propagation angle
!          at the top of the PE region at each range R.
!
! Version Number 1.3.0
!
! INPUTS:
!   Argument List: R, RLAST
!   Common: AATZ, DELZ, DR, FTER, ISM2, IZ, IZMAX, RATZ, RLOG, RLOGLST, YCUR,
!           YLAST, ZLIM
!   Public: U(), ULST()
!
! OUTPUTS:
!   Argument List: NONE
!   Common: IZ
!   Public: FFACZ(,)
!
! Modules Used: APM_MOD
!
! Calling Routines: PESTEP
!
! Routines called:
!   APM Specific: GETPFAC(function), SAVEPRO, SPECEST
!   Intrinsic: MIN0, DABS, DMIN1, DSIGN
!
! GLOSSARY: See universal glossary for common variables.
!
!   Input Variables:
!     R = Current PE range in meters.
!     RLAST = Previous PE range in meters.
!
!   Output Variables: See universal glossary for common variables.
!
!   Local Variables:
!     ANGDIFF = The difference between current outgoing propagation
!               angle and previous angle determined.
!     IZP = Previous index counter in FFACZ().
!     PFDB = Propagation factor in dB at current PE range R at height
!            ZLIM.
!     PFDBLST = Propagation factor in dB at previous PE range RLAST at height
!              ZLIM.
!     PFRATZ = Propagation factor in dB at range RATZ and height ZLIM.
!     THOUT = Outgoing propagation angle determined at top of PE region.

subroutine fzlim( r, rlast )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

! FFACZ(I,1) = propagation factor in dB
! FFACZ(I,2) = range in meters
! FFACZ(I,3) = angle in radians

pfdb = getpfac( u, rlog, delz, zlim-ycur )

if( iz .eq. 1 ) then
  pfdblst = getpfac( ulst, rloglst, delz, zlim-ylast )
  frac = ( ratz - rlast ) / dr
  pfratz = pfdblst + frac * ( pfdb - pfdblst )
```

```

        ffacz(iz,1) = pfratz
        ffacz(iz,2) = ratz
        ffacz(iz,3) = aatz
        call savepro
    end if

    ! Perform spectral estimation using top layer from height=JZLIM*DELZ to
    ! height=(JZLIM-NPNTS)*DELZ. Determine outgoing propagation angle THOUT.

    call specest( 0, THOUT )

    iz = iz + 1
    iz = min0( iz, izmax )
    ffacz(iz,1) = pfdb
    ffacz(iz,2) = r

    ! Do not let THOUT become greater than the GOOD portion of the maximum
    ! PE propagation angle if less than ISM2 PE range steps from start of
    ! X0 region.

    ffacz(iz,3) = thout
    if( iz .le. ism2 ) ffacz(iz,3) = dmin1( aatz, thout )

    if( iz .ge. 2 ) then

        ! To avoid extreme "spiking", limit the change in angle values.

        izp = iz - 1
        if( .not. fter ) then
            ffacz(iz,3) = dmin1( ffacz(izp,3), thout )
            angdif = ffacz(iz,3) - ffacz(izp,3)
            if( dabs(angdif) .gt. 1.d-4 )
                ffacz(iz,3) = ffacz(izp,3) + dsign(1.,angdif)*1.d-4
            &
        else
            if( iz .le. 10 ) then
                angdif = ffacz(iz,3) - ffacz(izp,3)
                if( dabs(angdif) .gt. 1.d-4 )
                    ffacz(iz,3) = ffacz(izp,3) + sign(1., angdif)*1.d-4
                &
            end if
        end if

    end if

    call savepro

end subroutine fzlim

```

A.2.9 Function GETPFAC

```

! ***** FUNCTION GETPFAC *****

! Module Name: GETPFAC

! Module Security Classification: UNCLASSIFIED

! Purpose: Performs linear interpolation in height on magnitude of the
!          PE field and then calculates propagation factor in dB.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: DELZ, HEIGHT, RLOG
!   Common: NONE
!   Public: U()

! OUTPUTS:
!   Function: GETPFAC

! Calling Routines: CALCLOS, FZLIM

```

```

! Routines Called:
!   APM Specific: NONE
!   Intrinsic: CDABS, DBLE, DLOG10, DMAX1, IDINT

! GLOSSARY:

!   Input Variables:
!       DELZ = Bin width in z-space = WL / (2*sin(THETAMAX))
!       RLOG = 10. * alog10( PE range )
!       HEIGHT = receiver height in meters
!       U() = Complex array containing PE field solution.

!   Output Variables:
!       GETPFAC = Propagation factor at height HEIGHT in dB.

!   Local Variables:
!       FB = Real number of bins corresponding to HEIGHT.
!       FR = Real difference between FB and NB.
!       NB = Integer number of bins corresponding to HEIGHT.
!       NBP1 = NB + 1
!       U0 = Complex field at bin directly below (NB) desired height HEIGHT.
!       U1 = Complex field at bin directly above (NBP1) desired height HEIGHT.
!       PMAG0 = Magnitude of field at bin NB.
!       PMAG1 = Magnitude of field at bin NBP1.
!       PMAG = Interpolated magnitude.
!       PMAGMIN = Lower limit on magnitude of field to avoid underflow/
!                 overflow problems.
!       GETPFAC = Propagation factor squared in dB [i.e. 20*log10(F)].

real(kind=8) function GETPFAC( u, rlog, delz, height )

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

complex(kind=8) u(0:*), u0, u1

data pmagmin/1.d-12/

fb = height / delz
nb = idint(fb)
fr = fb - dble(nb)
nbp1=nb+1

u0=u(nb)
u1=u(nbp1)

pmag0 = cdabs( u0 )
pmag1 = cdabs( u1 )

pmag = pmag0 + fr * (pmag1 - pmag0)

pmag = dmax1( pmag, pmagmin )
getpfac = 20.*dlog10( pmag ) + rlog

end function getpfac

```

A.2.10 Subroutine GETREFCOEF

```

! ***** SUBROUTINE GETREFCOEF *****

! Module Name: GETREFCOEF

! Module Security Classification: UNCLASSIFIED

! Purpose: Calculates the complex reflection coefficient.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: ANGLE, IFLAG, RANGE

```

```

!   Common: IGR, IPOL, NW, RUF, RUF_FAC
!   Public: CN2(), RGRND(), RNGWIND(), WIND()
!   Parameter: PI

!   OUTPUTS:
!   Argument List: REFCOEF, RMAG, RPHASE
!   Common: RUF_HT

!   Modules Used: APM_MOD

!   Calling Routines: FEM, GETALN, ROCALC

!   Routines called:
!   APM Specific: NONE
!   Intrinsic: CDABS, CDSQRT, CMPLX, DATAN2, DCOS, DIMAG, DSIN, DSQRT, MAX0, REAL

!   GLOSSARY: See universal glossary for common variables and parameters.

!   Input Variables:
!   ANGLE = grazing angle
!   IFLAG = Integer flag indicating if reflection calculation is being
!           performed within FE and RO regions (IFLAG=0) or PE region
!           (IFLAG=1).
!   RANGE = range in meters of reflection point(IFLAG=0) or PE
!           range(IFLAG=1)

!   Output Variables:
!   REFCOEF = complex reflection coefficient
!   RMAG = magnitude of the reflection coefficient
!   RPHASE = phase of the reflection coefficient

!   Local Variables:
!   CRAD = Term used in calculation of reflection coefficient.
!           CRAD = sqrt[ n**2 - (cos(angle))**2 ] where n = index
!           of refraction.
!   REFC = Complex reflection coefficient for circular polarization.
!   REFH = Complex reflection coefficient for horizontal polarization.
!   REFV = Complex reflection coefficient for vertical polarization.
!   RFRAC = Fractional range for interpolation on wind speed
!   RNG2T = Complex dielectric constant applied at the point of reflection.
!   RUFFAC = Rough surface reduction factor.
!   SRAD = Term used in calculation of reflection coefficient.
!           SRAD = n**2 * sin(angle) where n=index of refraction.
!   WND = Interpolated wind speed

subroutine getrefcoef( iflag, angle, range, REFCOEF, RMAG, RPHASE )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

complex(kind=8) refcoef, crad, srad, rng2t, refv, refc, refh

refv = cmplx(0., 0., 8)
refh = cmplx(0., 0., 8)
refc = cmplx(0., 0., 8)

if( igr .eq. 1 ) then
    rng2t = cn2(1)
else
    if( iflag .eq. 0 ) then      !Called for RO or FE calcs
        k = 1
        do while(( rgrnd(k) .lt. range ) .and. ( k .lt. igr ))
            k = k + 1
        end do
        k = max0( 1, k-1 )
        rng2t = cn2(k)
    else
        rng2t = cn2(igr)      !Called for PE calcs
    end if
end if

```

```

end if

if( ipol .eq. 1 ) then

! Compute complex reflection coefficient for vertical polarization.

ctheta = dcos( angle )
stheta = dsin( angle )
crad = cdsqrt( rng2t - ctheta*ctheta )
srad = rng2t * stheta
refv = (srad - crad) / (srad + crad)
refcoef = refv

end if

if(( ipol .eq. 2 ) .or. ( ipol .eq. 0 )) then

! Compute complex reflection coefficient for horizontal polarization.

ctheta = dcos( angle )
stheta = dsin( angle )
crad = cdsqrt( rng2t - ctheta*ctheta )
refh = (stheta - crad) / (stheta + crad)
refcoef = refh

end if

! Compute complex reflection coefficient for circular polarization.

if( ipol .eq. 2 ) then
refc = .5 * ( refv + refh )
refcoef = refc
end if

! Compute rough surface reduction factor and multiply by reflection
! coefficient.

if( ruf ) then

! Determine proper wind speed for current range.

if( nw .gt. 1 ) then
k = 1
do while(( range .gt. rngwind(k+1) ) .and. ( k .lt. nw ))
k = k + 1
end do
if( range .gt. rngwind(nw) ) then
ruf_ht = ruf_fac * wind(nw)**2
else
rfrac = (range - rngwind(k)) / (rngwind(k+1)-rngwind(k))
wnd = wind(k) + rfrac * ( wind(k+1) - wind(k) )
ruf_ht = ruf_fac * wnd**2
end if
end if

g = ruf_ht * dsin( angle )
xg = .5 * g**2
xgc = 3.2 * xg
ruffac = 1. / dsqrt(xgc - 2.d0 + dsqrt(xgc**2 - 7.d0 * xg + 9.d0))
refcoef = refcoef * ruffac
end if

rmag = cdabs( refcoef )
rphase = datan2( dimag( refcoef ), real( refcoef, 8 ) )

end subroutine getrefcoef

```

A.2.11 Subroutine MIXEDFT

```

! ***** SUBROUTINE MIXEDFT *****

```

```

! Module Name: MIXEDFT

! Module Security Classification: UNCLASSIFIED

! Purpose: Propagates the PE field in free space one PE range step and
!          applies the Leontovich boundary condition using the discrete
!          mixed Fourier transform as outlined by Dockery and Kuttler in
!          "An Improved Impedance-Boundary Algorithm for Fourier Split-
!          Step Solutions of the Parabolic Wave Equation", IEEE Trans.
!          on Ant. & Prop., Vol. 44, No. 12, Dec. 1996, pp. 1592-1599, for
!          the central difference algorithm. Also uses the backward difference
!          algorithm as described in APL report A2A-00-U-0-010, 8 August 2000.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: None
!   Common: ALPHAQ, C1X, C2X, CK1, CK2, CMFT, CMFT_X, DZ2, IALG, N, NM1, RK, RT
!   Public: RN(), U()

! OUTPUTS:
!   Argument List: None
!   Common: CK1, CK2, CMFT
!   Public: U(), W(), YM()

! Modules used: APM_MOD

! Calling Routines: PESTEP

! Routines Called:
!   APM Specific: FRSTP
!   Intrinsic: CMPLX, SUM

! GLOSSARY:

!   Input Variables: See universal glossary for common variables.

!   Output Variables: See universal glossary for common variables.

!   Local Variables:
!     AR = Complex coefficient of partial linear solution to homogeneous equ.
!     BR = Complex coefficient of partial linear solution to homogeneous equ.
!     ARX = Partial linear solution to homogeneous equ.
!     BRX = Partial linear solution to homogeneous equ.
!     C1C = Summation argument in determining AR.
!     C2C = Summation argument in determining BR.
!     SUM1 = Summation term in determining AR.
!     SUM2 = Summation term in determining BR.

subroutine mixedft(ipestp)

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

complex(kind=8) ar, arx, br, brx, c2c, sum1, sum2

w(0) = cmplx( 0., 0., 8 )
w(n) = cmplx( 0., 0., 8 )

if( ialg .eq. 1 ) then      !old central difference

  do i = 1, nm1
    w(i) = (u(i+1) - u(i-1)) / dz2 + alphaq * u(i)
  end do

! Transform W() to p-space, then multiply by free-space propagator,
! then transform back. Upon return W() is in z-space.

```

```

    call frstp( W )

! Propagate C1 and C2 coefficients to new range.

    ck1 = ck1 * c1x
    ck2 = ck2 * c2x
    ym(0) = cmplx(0.d0,0.d0,8)
    do i = 1, nm1
        ym(i) = dz2 * w(i) + rt * ym(i-1)
    end do

! Compute particular solution.

    u(n) = cmplx(0.d0,0.d0,8)
    do i = 1, N
        nmi = n - i
        u(nmi) = rt * (ym(nmi) - u(nmi+1))
    end do

!At this point U() is the particular solution.
!Determine coefficients AR and BR for homogeneous solution.

    sum1 = .5 * ( u(0) + u(n)*rn(n) )
    sum2 = .5 * ( u(0)*rn(n) + u(n) )

    do i = 1, nm1
        c2c = u(n-i) * rn(i) * (-1)**i
        sum2 = sum2 + c2c
    end do

    ar = ck1 - rk * ( sum1 + sum( u * rn ) )
    br = ck2 - rk * sum2

!Now compute total solution as the sum of the particular and
!homogeneous solutions.

    do i = 0, n
        arx = ar * rn(i)
        nmi = n-i
        brx = br * rn(nmi) * (-1)**(nmi)
        u(i) = u(i) + arx + brx
    end do

else                                     !Backward difference

    do i = 1, nm1
        w(i) = u(i) - rt * u(i-1)
    end do

! Transform W() to p-space, then multiply by free-space propagator,
! then transform back. Upon return W() is in z-space.

    call frstp( W )

! Propagate CMFT coefficient to new range.

    cmft = cmft * cmft_x

! Compute particular solution.

    u(0) = cmplx( 0., 0., 8 )
    do i = 1, N
        u(i) = w(i) + rt * u(i-1)
    end do

!At this point U() is the particular solution.
!Determine coefficient AR for homogeneous solution.

    ar = cmft - sum( u(0:nm1) * rn(0:nm1) )

!Now compute total solution as the sum of the particular and

```

!homogeneous solutions.

u = u + ar * rn

end if

end subroutine mixedft

A.2.12 Subroutine PESTEP

```
! ***** SUBROUTINE PESTEP *****
! Module Name: PESTEP
! Module Security Classification: UNCLASSIFIED
! Purpose: Propagates the PE field by one output range step DROUT.
! Version Number: 1.3.0
! INPUTS:
!   Argument List: ISTEP, ROUT
!   Common: DR, DR2, FTER, IALG, IG, IGR, IPE, IPOL, IXO, IZ, IZINC,
!           N, N34, NF4, NPROF, PEFLAG, RATZ, RLOG, RMAX, RUF, YCUR, YLAST
!   Public: FILT(), GRAZE(), PROFINT(), RGRND(), TYH(), U()
!   Saved: IPESTP, R, RLAST
!   Parameter: QI
! OUTPUTS:
!   Argument List: JEND, JSTART, MPFL
!   Common: IG, RLOG, RLOGLST, YCUR, YCURM, YLAST
!   Public: ENVPR(), U(), ULST()
!   Other: RLAST, RMID
!   Saved: IPESTP, R, RLAST
! Modules Used: APM_MOD
! Calling Routines: APMSTEP
! Routines Called:
!   APM Specific: CALCLOS, DOSHIFT, FRSTP, FZLIM, GETALN, REFINTER, MIXEDFT
!   Intrinsic: CDEXP, DABS, DLOG10, MIN0
! GLOSSARY: See universal glossary for common variables.
!   Input Variables:
!     ISTEP = Index of current output range step.
!     ROUT = Current output range in meters.
!   Output Variables:
!     JEND = Ending index within MPFL() of PE loss values.
!     JSTART = Starting index within MPFL() of PE loss values.
!     MPFL(,) = 2-byte integer array containing propagation factor(F) and loss
!               values in centibels vs. height, at each output range ROUT.
!               All values returned are referenced to height HMIN.
!               MPFL(1,i) = loss at output height equal to i*DZOUT
!               MPFL(2,i) = 20*log10(F) at output height equal to i*DZOUT
!   Local Variables:
!     IPESTP = Counter indicating current PE range step.
!     IZT = Counter used in order to determine when to compute outgoing
!           propagation angle, and save propagation factor and refractivity
!           profile.
!     R = Current PE range in meters.
!     RLAST = PE range at previous step in meters.
!     RMID = Range at which interpolation for range-dependent refractivity
!            profiles is performed. This is equal to the range midway
!            between the current and next PE range.
subroutine pestep( istp, rout, mpfl, jstart, jend )
```



```

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

integer(kind=2) mpfl(2,0:*)

save r, ipestp, rlast

! Initialize local variables.

if( istp .eq. 1 ) then
    r = 0.
    rlog = 0.
    izt = 0
    ipestp = 0
end if

! Begin loop.

DO while( r .lt. rout )

    if( r .gt. 0. ) ylast = ycur
    rlast = r
    rlogst = rlog

! Store the field arrays of the previous range step for subsequent horizontal
! interpolation at range ROUT.

    ulst = u

    r = r + dr
    ipestp = min0( ipestp + 1, ipe )
    rlog = 10. * dlog10( r )
    rmid = r - dr2

    if( fter ) then
        ycur = tyh(ipestp)

! Determine height at 1/2 range step - for interpolation on refractivity
! profiles.

        ycurm = .5 * ( tyh(ipestp-1) + ycur )

        if( ycur .lt. ylast ) call doshift
    end if

! If range-dependent and/or terrain case, then interpolate on profile. Compute
! z-space arrays.

    if(( nprof .gt. 1 ) .or. ( fter )) then
        call refinter( istp, rmid )

! Compute environmental phase term for 2nd part of PE step.

        envpr = cdexp( qi*dr * profint )

! Filter upper 1/4 of the array.

        envpr(n34:n) = filt(0:nf4) * envpr(n34:n)
    end if

! Adjust array index for dielectric parameter, if necessary, and calculate
! impedance term for smooth surface and non-horizontal polarization.

    if(( ipol .eq. 1 ) .and. ( ig+1 .le. igr )) then
        if( r .gt. rgrnd(ig+1) ) ig = ig + 1
        if( .not. ruf ) call getaln( graze(0), r )
    end if

```

```

! If performing a rough surface case then calculate impedance term.
    if(( ruf ) .and. ( ycur .lt. 1.d-3 )) call getaln( graze(ipestp), r )

    if( ialg .gt. 0 ) then
        call mixedft(ipestp)
    else
        call frstp(u)
    end if

! Multiply by environment term.

    u = u * envpr

    if(( fter ) .and. ( ycur .ge. ylast )) call doshift

! Store propagation factor along with current range and outgoing
! propagation angle if using hybrid method (for extended optics).

    if((( ixo .ge. 1 ) .and. ( r .gt. ratz )) .and. (.not. peflag)) then
        if( iz .eq. 1 ) call fzlim( r, rlast )
        izt = izt + 1
        if(( izt .eq. izinc ) .or. ( dabs(r-rmax) .lt. dr )) then
            call fzlim( r, rlast )
            izt = 0
        end if
    end if

end do

! Calculate propagation loss at range ROUT.

call calclos( rlast, istp, MPFL, JSTART, JEND )

end subroutine pestep

```

A.2.13 Subroutine RAYTRACE

```

! ***** SUBROUTINE RAYTRACE *****

!   AUTHOR:
!       Herb Hitney
!       Space and Naval Warfare Systems Center, San Diego
!       Tropospheric Branch, Code D883
!   ADDRESS:
!       SPAWARSYSCEN SAN DIEGO D883
!       49170 PROPAGATION PATH
!       SAN DIEGO CA 92152-7385
!       Tel: 619-553-1428 DSN 553-1428
!       Fax: 619-553-1417 DSN 553-1417
!       Email: herb@spawar.navy.mil

! With minor code modifications by:
! Author: Amalia E. Barrios
!       SPAWARSYSCEN SAN DIEGO D858
!       49170 Propagation Path
!       San Diego, CA 92152-7385
!       e-mail: barrios@spawar.navy.mil
!       phone: (619) 553-1429
!       fax: (619) 553-1417

!Module Name: RAYTRACE

!Module Security Classification: UNCLASSIFIED

! PURPOSE: Computes full raytrace to range ROUT for elevation
!          angle at source height.

!Version Number: 1.3.0 modified from

```

```

!               RPO 1.15B      DATE:  19 August 1996

!INPUTS:
!  Argument list: A, ROUT
!  Common: ISTART, LEVELS
!  Public: GR(), Q(), RM(), ZRT()

!OUTPUT:
!  Argument list: AB, DXDA, ITYPE, PLD, PSI, XREFLECT, ZR

!Modules Used: APM_MOD

!CALLING ROUTINES: ROCALC

!ROUTINES CALLED:
!  APM Specific: NONE
!  Intrinsic: DABS, DSIGN, DSQRT

!GLOSSARY: See universal glossary for common variables.

!  Input Variables:
!    A = elevation angle at source in radians
!    ROUT = terminal range in meters

!  Output Variables:
!    AB = elevation angle at end of ray step in radians
!    DXDA = derivative of ROUT w.r.t. a in meters per radian
!    ITYPE = 0 for direct ray, 1 for reflected ray
!    PLD = optical path length difference from ROUT in meters
!    PSI = 0 for direct ray, grazing angle for ref. ray in radians
!    XREFLECT = Range at which ray is reflected in R0 and FE calculations.
!    ZR = terminal height in meters

!  Local Variables:
!    AA = elevation angle at start of ray step in radians
!    DELX = range increment in one ray trace step in meters
!    DELZR = height increment in one ray trace step in meters
!    GOFLAG = logical flag, true normally, false to stop raytrace
!    RAD = radical for square root test in ray trace step
!    XSUM = running sum of range during ray trace in meters
!    XTEMP = temporary range in ray trace step in meters
!    ZLIMIT = maximum height of ray that turns around in meters
!    ZMIN = minimum height of ray that turns around in meters

SUBROUTINE raytrace (rout, a, ZR, AB, DXDA, PLD, PSI, XREFLECT, ITYPE)

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

LOGICAL goflag

! Set initial conditions at start of ray.

xreflect = 0.
aa = a
i = istart
xsum = 0.
dxda = 0.
pld = 0.
psi = 0.
itype = 0
goflag = .TRUE.

! Main loop repeats until goflag is false (XSUM = ROUT).

DO WHILE (goflag)

    aa2 = aa**2
    IF (aa .GE. 0.) THEN

```

```

    gri = gr(i)
    gri2 = 2. * gri

! Ray is upgoing.

    IF (i .EQ. levels) THEN

! Upgoing ray is in highest layer (last step).

        delx = rout - xsum
        ab = aa + delx * gri
        ab2 = ab**2
        delzr = (ab2 - aa2) / gri2
        zr = zrt(i) + delzr
        dxda = dxda + (a / ab - a / aa) / gri
        pld = pld + ((rm(i) - aa2 / 2.) * (ab - aa) +
            (ab2 * ab - aa2 * aa) / 3.) / gri &
        goflag = .FALSE.

    ELSE

! Upgoing ray is not in highest layer.

        rad = aa2 + q(i)
        IF (rad .GE. 0.) THEN

! Upgoing ray penetrates current layer.

            ab = dSQRT(rad)
            ab2 = ab**2
            delx = (ab - aa) / gri
            xtemp = xsum + delx
            IF (xtemp .LT. rout) THEN

! Full upgoing step in current layer.

                xsum = xtemp
                dxda = dxda + (a / ab - a / aa) / gri
                pld = pld + ((rm(i) - aa2 / 2.) * (ab - aa) +
                    (ab2 * ab - aa2 * aa) / 3.) / gri &
                aa = ab
                aa2 = aa**2
                i = i + 1
                gri = gr(i)
                gri2 = 2. * gri

            ELSE

! Final upgoing step in current layer.

                delx = rout - xsum
                ab = aa + delx * gri
                ab2 = ab**2
                zr = zrt(i) + (ab2 - aa2) / gri2
                dxda = dxda + (a / ab - a / aa) / gri
                pld = pld + ((rm(i) - aa2 / 2.) * (ab - aa) +
                    (ab2 * ab - aa2 * aa) / 3.) / gri &
                goflag = .FALSE.

            END IF

        ELSE

! Upgoing ray turns around in current layer.

            delx = -aa / gri
            xtemp = xsum + delx
            IF (xtemp .LT. rout) THEN

! Full step in upgoing segment.

```

```

        xsum = xtemp
        xtemp = xsum + delx
        IF (xtemp .LT. rout) THEN

! Full step in downgoing segment.

        xsum = xtemp
        ab = -aa

        ELSE

! Last step in downgoing segment.

        zlimit = zrt(i) - aa ** 2 / gri2
        delx = rout - xsum
        ab = delx * gr(i)
        delzr = ab ** 2 / gri2
        zr = zlimit + delzr
        goflag = .FALSE.

        END IF

        ELSE

! Last step in upgoing segment.

        delx = rout - xsum
        ab = aa + delx * gri
        zr = zrt(i) - (aa ** 2 - ab ** 2) / gri2
        goflag = .FALSE.

        END IF

! Following section applies to all upgoing rays that turn around.

        ab2 = ab**2
        dxda = dxda + (a / ab - a / aa) / gri
        pld = pld + ((rm(i) - aa2 / 2.) * (ab - aa) + &
            (ab2 * ab - aa2 * aa) / 3.) / gri
        aa = ab

        END IF
    END IF
ELSE

! Ray is downgoing.

    grim1 = gr(i-1)
    grim12 = 2. * grim1
    rad = aa2 - q(i - 1)

    IF (rad .GE. 0.) THEN

! Downgoing ray penetrates current layer.

        ab = -dSQRT(rad)
        delx = (ab - aa) / grim1
        xtemp = xsum + delx
        IF (xtemp .LT. rout) THEN

! Full downgoing step in current layer.

            xsum = xtemp
            dxda = dxda + (a / ab - a / aa) / grim1
            pld = pld + ((rm(i) - aa2 / 2.) * (ab - aa) + &
                (ab ** 3 - aa2 * aa) / 3.) / grim1
            aa = ab
            i = i - 1
            IF (i .EQ. 0) THEN

```

```

! Downgoing ray reflects from sea surface.

      itype = 1
      psi = dABS(aa)
      xreflect = xtemp
      xtemp = 2. * xsum
      IF (xtemp .LT. rout) THEN

! Use symmetry concept to double ray path up to source level.

      aa = -a
      i = istart
      xsum = xtemp
      dxda = 2. * dxda
      pld = 2. * pld

      ELSE

! Downgoing ray reflects, but symmetry concept is not used.

      aa = -aa

      END IF
      END IF
      aa2 = aa**2

      ELSE

! Final downgoing step in current layer.

      delx = rout - xsum
      ab = aa + delx * grim1
      zr = zrt(i) - (aa2 - ab ** 2) / grim12
      dxda = dxda + (a / ab - a / aa) / grim1
      pld = pld + ((rm(i) - aa2 / 2.) * (ab - aa) + &
        (ab ** 3 - aa2 * aa) / 3.) / grim1
      goflag = .FALSE.

      END IF
      ELSE

! Downgoing ray turns around in current layer.

      delx = -aa / grim1
      xtemp = xsum + delx
      IF (xtemp .LT. rout) THEN

! Full step in downgoing segment.

      xsum = xtemp
      xtemp = xsum + delx
      IF (xtemp .LT. rout) THEN

! Full step in upgoing segment.

      xsum = xtemp
      ab = -aa

      ELSE

! Last step is in upgoing segment.

      zmin = zrt(i) - aa2 / grim12
      delx = rout - xsum
      ab = delx * gr(i - 1)
      delzr = ab ** 2 / grim12
      zr = zmin + delzr
      goflag = .FALSE.

      END IF
      ELSE

```

```

! Last step is in downgoing segment.

      delx = rout - xsum
      ab = aa + delx * grim1
      delzr = (aa2 - ab ** 2) / grim12
      zr = zrt(i) - delzr
      goflag = .FALSE.

      END IF

! Following section applies to all downgoing rays that turn around.

      dxda = dxda + (a / ab - a / aa) / grim1
      pld = pld + ((rm(i) - aa2 / 2.) * (ab - aa) + &
        (ab ** 3 - aa2 * aa) / 3.) / grim1
      aa = ab

      END IF
    END IF
  END DO

! Terminal elevation angle ab cannot be zero.

IF (dABS(ab) .LT. 1.d-10) ab = dSIGN(1.d-10, ab)

END subroutine raytrace

```

A.2.14 Subroutine REFINTER

```

! ***** SUBROUTINE REFINTER *****

! Module Name: REFINTER

! Module Security Classification: UNCLASSIFIED

! Purpose: Interpolates vertically and horizontally on the refractivity profiles.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: ISTEP, RANGE
!   Common: FTER, HMINTER, IS, LVL, NPROF, RV2, YCURM
!   Public: HMSL(), REFMSL(), RNGPROF()

! OUTPUTS:
!   Argument List: NONE
!   Common: IS, LVLEP, RV2
!   Public: HTDUM(), PROFINT(), REFDUM()

! Modules Used: APM_MOD

! Calling Routines: GETGRAZE, PESTEP

! Routines Called:
!   APM Specific: INTPROF, REMDUP, PROFREF
!   Intrinsic: NONE

! GLOSSARY: See universal glossary for common variables.

!   Input Variables:
!     ISTEP = Current output range step index.
!     RANGE = Range for profile interpolation.

!   Output Variables: NONE

!   Local Variables:
!     ICHK = Used to set REFDUM() and HTDUM() to the last input profile
!           specified by REFMSL(NPROF), HMSL(NPROF) if RANGE is
!           beyond range of last input profile.

```

```

!      RV1 = Range of previous user-input profile.

subroutine refinter( istp, range )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

save j, rv1, ichk
data j, rv1 / 0, 0. /

if( istp .eq. 1 ) ichk = 0
if( (.not. fter ) .and. ( ichk .eq. 1 )) return
lvlep = lvlp

! If there is a range-dependent refractivity profile then interpolate horizontally
! using the two surrounding profiles at range RANGE with all duplicate levels.

if(( nprof .gt. 1 ) .and. ( ichk .eq. 0 )) then
  if( range .gt. rngprof( nprof ) ) then
    ichk = 1
    do i = 0, lvlep
      refdum(i) = refmsl(i,nprof)
      htdum(i) = hmsl(i,nprof)
    end do
  else
    IF( range .gt. rv2 ) then
      j = is
      IS=IS+1
      rv1=rv2
      rv2=rngprof(IS)
    end if

    FV=(range-rv1)/(rv2-rv1)

    do i = 0, lvlep
      refdum(i) = plint( refmsl(i,j), refmsl(i,is), fv )
      htdum(i) = plint( hmsl(i,j), hmsl(i,is), fv )
    end do
  end if

! Now remove all duplicate levels with LVLEP now being the # of points in the
! profile at range RANGE.

  call remdup
  call profref( hminter, 0 )

! At this point REFNUM() and HTDUM(), also HREF() and REFREF(), are referenced
! to HMINTER.

end if

! Using BS method must determine height and M-unit profiles relative to ground,
! where YCURM is now the height of the local ground above the reference height
! HMINTER.

call profref( ycurm, 1 )

! Interpolate vertically with height. PROFINT is now an N-point (N=2**NFFT)
! array containing the interpolated M-unit values for the refractivity at
! range RANGE.

call intprof

end subroutine refinter

```

A.2.15 Subroutine ROCALC

```

! ***** SUBROUTINE ROCALC *****

```



```

!   AUTHOR:
!       Herb Hitney
!       Space and Naval Warfare Systems Center San Diego
!       Atmospheric Propagation Branch, Code D858
!   ADDRESS:
!       SPAWARSYSCEN SAN DIEGO D858
!       49170 PROPAGATION PATH
!       SAN DIEGO CA 92152-7385
!       Tel: 619-553-1428 DSN 553-1428
!       Fax: 619-553-1417 DSN 553-1417
!       Email: herb@spawar.navy.mil

! With minor code modifications by:
! Author: Amalia E. Barrios
!       SPAWARSYSCEN SAN DIEGO D858
!       49170 Propagation Path
!       San Diego, CA 92152-7385
!       e-mail: barrios@spawar.navy.mil
!       phone: (619) 553-1429
!       fax: (619) 553-1417

! Module Name: ROCALC

! Module Security Classification: UNCLASSIFIED

! PURPOSE: Computes and stores ray-optics components as needed
! to "span" range X. Arrays use the index K, where the elevation
! angle in radians at the origin is GAMMA = K/1000. XROP and XRON
! are the ranges less and greater than X, respectively. IROP and
! IRON are indices of the component arrays that correspond to XROP
! and XRON. The arrays are: DMAGSQ(,) and RMAGSQ(,) = the magnitude
! squared of the direct and reflected rays; and OMEGA(,) = phase angle
! in rad between direct and reflected rays. Ray-optics components
! are derived from calls to sub raytrace. Newton's method of
! iteration is used to find the direct and reflected elevation
! angles alphas & alphas. Parallel-ray approximations are used as
! starting values for the highest value of K, otherwise the most
! recent values of ALPHAD & ALPHAR are used to start the iteration.
! Note that KMIMP and KMINN are the minimum K values for good
! solutions at ranges XROP and XRON, and KMAX is the maximum K
! needed to exceed HTLIM at both XROP and XRON.

!Version Number: 1.3.0 modified from RPO 1.15B

! INPUTS:
!   Argument List: X
!   Common: ANTREF, BW, FKO, HTLIM, HTYDIF, IPOL, IRON, IROP, KMIMP, PSILIM, TWOKA,
!           XRON, XROP, YFREF, ZTOL
!   Public: ZOUTMA(), ZOUTPA(), ZRO()
!   Parameters: PI

! OUTPUTS:
!   Argument List: NONE
!   Common: ALPHAD, DELXRO, DMAGSQ(,), HTYDIF, IRON, IROP, KMAX, KMIMP,
!           KMIMP, OMEGA(,), RMAGSQ(,), XRON, XROP

! SAVE: DALPHA, FRACR0

! Modules used: APM_MOD

! CALLING ROUTINES: ROLOSS

! ROUTINES CALLED:
!   APM specific: ANTPAT, GETREFCOEF, RAYTRACE
!   Intrinsic: COUNT, DABS, DATAN, DMAX1, IDINT

! GLOSSARY: See universal glossary for common variables.

!   Input Variables:
!       X = Current output range in meters

```

```

!   Output Variables: NONE

!   Local Variables:
!       ALPHAR = reflected ray source elevation angle in radians
!       BETAD = direct ray terminal elevation angle in radians
!       BETAR = reflected ray terminal elevation angle in radians
!       DALPHA = one half the antenna beamwidth in radians
!       DXDAD = direct ray derivative of range w.r.t. elev angle
!       DXDAR = reflected ray derivative of range w.r.t. elev angle
!       DZDAD = direct ray derivative of height w.r.t. elev angle
!       DZDAR = reflected ray derivative of height w.r.t. elev angle
!       FRACRO = R0 range step fraction (0. to .25)
!       FSQD = propagation factor squared for direct ray
!       FSQR = propagation factor squared for reflected ray
!       GMAXDA = term used in computing R0 range step fraction
!       ITER = iteration counter (1 to 10)
!       ITYPE = ray type flag (0 = direct, 1 = reflected)
!       PFACD = antenna pattern factor for direct ray
!       PFACR = antenna pattern factor for reflected ray
!       PHI = phase lag of reflection coefficient in radians
!       PLDD = path length difference from x for direct ray
!       PLDR = path length difference from x for reflected ray
!       PSI = grazing angle in radians
!       REFCOEF = complex reflection coefficient
!       RMAG = magnitude of reflection coefficient
!       XREFLECT = Range at which ray is reflected in R0 and FE calculations.
!       ZD = terminal height of direct ray in meters
!       ZK = height of kth R0 index in meters
!       ZR = terminal height of reflected ray in meters

```

```

SUBROUTINE rocalc(x)

```

```

  use apm_mod

```

```

  implicit integer(kind=4) (i-n)
  implicit real(kind=8) (a-h, o-z)

```

```

  complex(kind=8) refcoef

```

```

  SAVE dalpha, fracRO

```

```

  data deg / .01745d0 /    !1 degree in radians

```

```

  ! Test if new R0 calculations are needed. First time is indicated
  ! by IROP = -1

```

```

DO WHILE (x .GE. xR0n)
  IF (iROP .EQ. -1) THEN
    iROP = 1
    iR0n = 0
    xR0n = x
    kmax = 88
    kminp = 0
    kminn = 0
    fracRO = 0.
    dalpha = bw / 2.
    htydif = htlim - yfref
    IF (dalpha .GT. deg) dalpha = deg
  ELSE
    xR0p = xR0n
    iROP = 1 - iROP
    iR0n = 1 - iR0n
    kminp = kminn
    kminn = 0
    kmax = idINT(1000. * htydif / xR0p) + 2
    IF (kmax .GT. 88) kmax = 88
    IF (fracRO .LT. .25) THEN
      gmaxda = dMAX1((.001 * kmax) / dalpha, 5.0)
      fracRO = 1. / (gmaxda - 1.)
    END IF
  END IF

```

```

        delxR0 = fracR0 * xR0p
        xR0n = xR0p + delxR0
    END IF

! Set starting conditions corresponding to highest angle.
! Assume parallel direct & reflected rays to start. Note DZDAD
! and DZDAR are the direct and reflected ray derivatives of
! height w.r.t. elevation angle at the source.

    alphad = .001 * kmax
    alphas = -alphad
    CALL raytrace(xR0n, alphad, ZD, BETAD, DXDAD, PLDD, PSI, XREFLECT, ITYPE)
    dzdad = -betad * dxdad
    CALL raytrace(xR0n, alphas, ZR, BETAR, DXDAR, PLDR, PSI, XREFLECT, ITYPE)
    dzdar = -betar * dxdar

! Main loop to compute all R0 components at height ZK.

    k = kmax
    DO WHILE (k .GE. kminn)
        IF (k .GT. 0) THEN
            zk = xR0n * .001 * k
        END IF

! Loop to find direct ray and components at ZK.

        iter = 0
        DO WHILE (iter .LT. 10)
            iter = iter + 1
            alphad = alphad - (zd - zk) / dzdad
            CALL raytrace(xR0n, alphad, ZD, BETAD, DXDAD, PLDD, PSI, &
                XREFLECT, ITYPE)
            dzdad = -betad * dxdad
        END DO

! Test for direct ray not being found.

        IF ((dABS(dzdad) .LT. 1.d-6) .OR. (itype .EQ. 1)) THEN
            kminn = k + 1
            iter = 10
        END IF

! Test for convergence of direct ray.

        IF (dABS(zk - zd) .LT. ztol) iter = 10
    END DO

! Loop to find reflected ray and components at ZK.

    iter = 0
    DO WHILE (iter .LT. 10)
        iter = iter + 1
        alphas = alphas - (zr - zk) / dzdar
        CALL raytrace(xR0n, alphas, ZR, BETAR, DXDAR, PLDR, PSI, &
            XREFLECT, ITYPE)
        dzdar = -betar * dxdar
    END DO

! Test for reflected ray not being found.

    IF ((dABS(dzdar) .LT. 1.d-6) .OR. (itype .EQ. 0)) THEN
        kminn = k + 1
        iter = 10
    END IF

! Test for convergence of reflected ray.

    IF (dABS(zk - zr) .LT. ztol) iter = 10
END DO

! Test for grazing angle less than limiting value.

    IF (psi .LT. psilim) kminn = k

```

```

! Compute magnitude of direct and reflected rays
! based on ray focusing.

      fsqd = dABS(xRON / dzdad)
      fsqr = dABS(xRON / dzdar)

! Adjust magnitude of direct and reflected rays based on
! antenna patterns and reflection coefficient.

      CALL antpat( alphad, PFACD )
      CALL antpat( alphas, PFACR )

      CALL getrefcoef( 0, psi, xreflect, REFCOEF, RMAG, RPHASE )
      fsqd = fsqd * pfacd ** 2
      fsqr = fsqr * (pfacr * rmag) ** 2

! Store ray-optics components in proper arrays. Phase lag, OMEGA(,),
! is computed based on total path length difference of the two rays
! plus the reflection coefficient phase lag, PHI.

      dmagsq(iRON, k) = fsqd
      rmagsq(iRON, k) = fsqr
      omega(iRON, k) = (pldr - pldd) * fko + rphase

! Force field to zero at the surface by making magnitudes equal and phase
! lag PI for H pol. For V pol force R0 field to go to field based on
! FE calculations.

      ELSE

      dmagsq(iRON, 0) = fsqd
      rmagsq(iRON, 0) = fsqr
      omega(iRON, 0) = -pi
      if( ipol .eq. 1 ) then      !V pol
        rx2 = xron**2
        imns = count( zro .lt. 0.d0 )
        zm = zoutma(imns) - rx2 / twoka
        zp = zoutpa(imns) - rx2 / twoka
        xreflect = xron * antref / zp

! ALPHAD = direct ray angle
! ALPHAR = reflected ray angle (grazing angle = -ALPHAR)

        alphad = datan( zm / xron )
        alphas = datan( zp / xron )

        call antpat( alphad, FACD )
        call antpat( -alphas, FACR )

! Determine reflection coefficient.

        call getrefcoef( 0, alphas, xreflect, REFCOEF, RMAG, RPHASE )

        dmagsq(iRON, 0) = facd**2
        rmagsq(iRON, 0) = (facr*rmag)**2
        omega(iRON, 0) = rphase
      end if
    END IF

! Decrement K index.

      k = k - 1

    END DO

! End of loop that advances ray optics solution to XRON.

  END DO

END subroutine rocalc

```

A.2.16 Subroutine ROLOSS

```
! ***** SUBROUTINE ROLOSS *****
!
!   AUTHOR:
!       Herb Hitney
!       Space and Naval Warfare Systems Center San Diego
!       Tropospheric Branch, Code D883
!   ADDRESS:
!       SPAWARSYSCEN SAN DIEGO D883
!       49170 PROPAGATION PATH
!       SAN DIEGO CA 92152-7385
!       Tel: 619-553-1428 DSN 553-1428
!       Fax: 619-553-1417 DSN 553-1417
!       Email: herb@spawar.navy.mil
!
! With minor executable code modifications by:
! Author: Amalia E. Barrios
!       SPAWARSYSCEN SAN DIEGO D858
!       49170 Propagation Path
!       San Diego, CA 92152-7385
!       e-mail: barrios@spawar.navy.mil
!       phone: (619) 553-1429
!       fax: (619) 553-1417
!
!Module Name: ROLOSS
!
!Module Security Classification: UNCLASSIFIED
!
! PURPOSE: Sets propagation loss in centibels at range ROUT for j from
! JMAX to JMIN based on 3 arrays obtained from sub ROCALC: DMAGSQ(,),
! RMAGSQ(,), and OMEGA(,). The 3 arrays are stored in order of (i,k),
! where i = 0 indicates components at range XROP (<ROUT), and i = 1
! indicates components at range XRON (>ROUT). K is the origin ray
! angle integer index in mrad [i.e. 1000 * angle]. KMINP and KMINN
! are the minimum good values of K at XROP and XRON, and KMAX is the
! maximum value of K where good components are stored at both XROP
! and XRON.
!
!Version Number: 1.3.0 modified from RPO 1.15B
!
! INPUTS :
!   Argument list: ISTEP, JMAX, JMIN, ROUT
!   Common: DELXRO, DMAGSQ(,), GASLOSS, IRON, IROP, KMAX, KMINN, KMINP, OMEGA(,),
!           RMAGSQ(,), XROP
!   Public: FSL(), ZRO()
!
! OUTPUTS:
!   Argument list: MPFL(,)
!   Common: NONE
!
! SAVE: DANGHI, DANGLO, DFSDHI, DFSILO, DFSRHI, DFSILO
!
! Modules Used: APM_MOD
!
! CALLING ROUTINES: APMSTEP, XOSTEP
!
! ROUTINES CALLED:
!   APM Specific: ROCALC
!   Intrinsic: DABS, DCOS, DLOG10, DMAX1, DSQRT, IDINT, IIDNNT
!
! GLOSSARY: See universal glossary for common variables.
!
!   Input Variables:
!       ISTEP = Current output range step index.
!       JMAX = Ending index within MPFL(,) of RO loss values.
!       JMIN = Starting index within MPFL(,) of RO loss values.
!       ROUT = Current output range in meters.
!
!   Output Variables:
```

```

!      MPFL(,) = 2-byte integer array containing propagation factor(F) and loss
!      values in centibels vs. height, at each output range ROUT.
!      All values returned are referenced to height HMIN.
!      MPFL(1,i) = loss at output height equal to i*DZOUT
!      MPFL(2,i) = 20*log10(F) at output height equal to i*DZOUT

!   Local Variables:
!   ANG = phase angle for computing FSQ in radians
!   ANGHI = phase angle above desired point in radians
!   ANGLO = phase angle below desired point in radians
!   DANGHI = diff. in phase angle along R0 step above desired point
!   DANGLO = diff. in phase angle along R0 step below desired point
!   DFSDHI = diff. in dir. mag**2 along R0 step above desired point
!   DFSDLO = diff. in dir. mag**2 along R0 step below desired point
!   DFSRHI = diff. in ref. mag**2 along R0 step above desired point
!   DFSRLO = diff. in ref. mag**2 along R0 step below desired point
!   FK = floating value of K index at jth output point
!   FFAC = propagation factor in dB
!   FSDHI = direct ray magnitude squared above desired point
!   FSDLO = direct ray magnitude squared below desired point
!   FSQ = propagation factor squared at desired point
!   FSQD = direct ray magnitude squared at desired point
!   FSRQ = reflected ray magnitude squared at desired point
!   FSRHI = reflected ray magnitude squared above desired point
!   FSRLO = reflected ray magnitude squared below desired point
!   KHI = K index above desired point
!   KLO = K index below desired point
!   KLOTMP = temporary KLO value
!   RATIOK = fraction of one K index (0. to 1.)
!   RATIOX = fraction of current R0 range step (0. to 1.)

SUBROUTINE roloss ( istp, rout, jmin, jmax, MPFL )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

INTEGER(kind=2) mpfl(2,0:*)

SAVE danghi, danglo, dfsdhi, dfsdlo, dfsrhi, dfsrlo

! Compute and store ray-optics components.

call rocalc( rout )

! Compute free-space loss term and ratio of distance from last R0
! range to R0 range increment. Set starting value of KLO to KMAX.

ratiox = (rout - xR0p) / delxR0
klo = kmax
fkro = 1000. / rout

! Loop to compute loss for all J from JMAX to JMIN.

DO j = jmax, jmin, -1

! Compute floating (non-integer) value of K corresponding to J, and
! integer value of K just below floating K. Test to see if this value
! is less than the previous value of KLO.

fk = fkro * zro(j)

klotmp = idINT(fk)

IF (klotmp .LT. klo) THEN

! Set new KLO and KHI.

klo = klotmp
khi = klo + 1

```

```

! If KLO is greater than or equal to the minimum K at range XROP
! and XRON, then compute new differences in components between
! XRON and XROP at index KLO. Otherwise old values will be used.

      IF ((klo .GE. kminp) .AND. (klo .GE. kminn)) THEN
        dfldo = dmagsq(iRON, klo) - dmagsq(iROP, klo)
        dfrlo = rmagsq(iRON, klo) - rmagsq(iROP, klo)
        danglo = omega(iRON, klo) - omega(iROP, klo)
      END IF

! If KHI is greater than or equal to the minimum K at range XROP
! and XRON, then compute new differences in components between
! XRON and XROP at index KHI. Otherwise old values will be used.

      IF ((khi .GE. kminp) .AND. (khi .GE. kminn)) THEN
        dfldhi = dmagsq(iRON, khi) - dmagsq(iROP, khi)
        dfrhi = rmagsq(iRON, khi) - rmagsq(iROP, khi)
        danghi = omega(iRON, khi) - omega(iROP, khi)
      END IF

! If KLO is greater than or equal to the minimum K at XROP, then
! compute new components at range ROUT at index KLO by linear inter-
! polation from range XROP at KLO. Otherwise interpolate backwards
! from XRON at KLO.

      IF (klo .GE. kminp) THEN
        fsldo = dmagsq(iROP, klo) + ratiox * dfldo
        frllo = rmagsq(iROP, klo) + ratiox * dfrlo
        anglo = omega (iROP, klo) + ratiox * danglo
      ELSE
        ratioxm1 = 1. - ratiox
        fsldo = dmagsq(iRON, klo) + ratioxm1 * dfldo
        frllo = rmagsq(iRON, klo) + ratioxm1 * dfrlo
        anglo = omega (iRON, klo) + ratioxm1 * danglo
      END IF

! If KHI is greater than or equal to the minimum K at XROP, then
! compute new components at range ROUT at index KHI by linear inter-
! polation from range XROP at KHI. Otherwise interpolate backwards
! from XRON at KHI.

      IF (khi .GE. kminp) THEN
        fsldhi = dmagsq(iROP, khi) + ratiox * dfldhi
        frlhi = rmagsq(iROP, khi) + ratiox * dfrhi
        anghi = omega (iROP, khi) + ratiox * danghi
      ELSE
        ratioxm1 = 1. - ratiox
        fsldhi = dmagsq(iRON, khi) + ratioxm1 * dfldhi
        frlhi = rmagsq(iRON, khi) + ratioxm1 * dfrhi
        anghi = omega (iRON, khi) + ratioxm1 * danghi
      END IF

END IF

ratiok = fk - klo
fsqd = fsldo + ratiok * (fsldhi - fsldo)
fsqr = frllo + ratiok * (frlhi - frllo)
ang = anglo + ratiok * (anghi - anglo)

! Compute square of propagation factor.

fsq = dABS(fsqd + fsqr + 2.d0 * dSQRT(dABS(fsqd * fsqr)) * dCOS(ang))

! Convert FSQ to propagation factor in dB. Limit to -250 dB.

ffac = 10. * dlog10( dmax1(1.d-25, fsq) )

! Compute and store propagation factor & loss in terms of closest
! integer centibel (cB). Note: gaseous absorption is included in the
! final propagation factor value that's output.

```

```

        dloss = fsl(istp) - ffac + gasloss
        ffac = fsl(istp) - dloss

        mpfl(1,j) = iidnnt( 10. * dloss )
        mpfl(2,j) = iidnnt( 10. * ffac )

END DO

END subroutine roloss

```

A.2.17 Subroutine SAVEPRO

```

! ***** SUBROUTINE SAVEPRO *****

! Module Name: SAVEPRO

! Module Security Classification: UNCLASSIFIED

! Purpose: Saves the refractivity profiles at each range step from the top of
!          the PE region to the maximum user-specified height. For use only
!          when using the hybrid model.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: NONE
!   Common: IZ, LVLEP, ZLIM
!   Public: HTDUM(), REFDUM()

! OUTPUTS:
!   Argument List: NONE
!   Common: NONE
!   Public: GRAD(,), HTR(,), LVL()

! Modules Used: APM_MOD

! Calling Routines: FZLIM

! Routines Called:
!   APM Specific: NONE
!   Intrinsic: DABS, DSIGN

! GLOSSARY: See universal glossary for common variables.

!   Input Variables: NONE

!   Output Variables: NONE

!   Local Variables:
!       G = Gradient of current refractivity profile level.
!       NEWL = Number of levels in refractivity profile from top of PE
!             region to maximum height.

subroutine savepro

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

! Determine at what index of current profile to begin storing from height
! ZLIM.

i = 0
do while( zlim .gt. ht dum(i) )
    i = i + 1
end do
i = i - 1
newl = -1

```



```

! Store gradients and height levels from this index level - I to LVLEP-1.
do j = i, lvlep-1
  jp1 = j + 1
  rm1 = reldum(j)
  rm2 = reldum(jp1)
  h1 = htdum(j)
  h2 = htdum(jp1)
  g = ( rm2 - rm1 ) / ( h2 - h1 )
  if( dabs( g ) .lt. 1.d-3 ) g = dsign( 1.d0, g )*1.d-3
  newl = newl + 1
  grad(newl,iz) = g * 1.d-6      ! for ray trace formulas
  htr(newl,iz) = h1
end do
newl = newl + 1
htr(newl,iz) = htdum(lvlep)
lvl(iz) = newl

end subroutine savepro

```

A.2.18 Subroutine SPECEST

```

!***** SUBROUTINE SPECEST *****

! Module Name: SPECEST

! Module Security Classification: UNCLASSIFIED

! Purpose: Determines the propagation angle THOUT based on spectral
!          estimation of either the topmost layer of the field still
!          within the "good" part of the transform, or the lower part
!          of the field (grazing angle), depending on the value of
!          IFLAG. For the upper portion of the field it looks at the
!          field from height=JZLIM*DELZ to height=(JZLIM-NPNTS)*DELZ.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: IFLAG
!   Common: DELZ, JZLIM, LNP, NP34, NPNTS, NSM1, XOCON, YCUR
!   Public: FILTP(), U()
!   Data: ISN

! OUTPUTS:
!   Argument List: THOUT
!   Common: NONE
!   Public: SPECTR(), XP(), YP()

! Modules Used: APM_MOD

! Calling Routine: FZLIM, GETGRAZE

! Routines Called:
!   APM Specific: DRST(in module APM_MOD)
!   Intrinsic: DASIN, DBLE, DIMAG, DLOG10, DMAX1, DSQRT, IDNINT, MAX0, MIN0, REAL

! GLOSSARY: See universal glossary for common variables.

!   Input Variables:
!     IFLAG = Integer flag indicating if spectral estimation is to be
!             performed on lower PE field (IFLAG=1, rough surface) or
!             upper PE field (IFLAG=0, X0).

!   Output Variables:
!     THOUT = outward propagation angle in radians at top of PE height
!             region.

!   Local Variables:
!     AMP = Field magnitude with lower limit of 1.e-10.

```

```

!      ATTN = Filter factor - used for filtering field before transforming.
!      IPEAK = Bin # in SPECTR() corresponding to the peak magnitude.
!      K = Bin # at which to start storing PE field.  Points from K to
!           K-NPNTS are stored in XP() and YP().
!      PAVG = 3-pt average magnitude.
!      PEAK = Peak magnitude.
!      PP = Field magnitude.

subroutine specest( iflag, THOUT )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

if( iflag .eq. 0 ) then

! Store upper NPNTS of U() in XP() and YP().  This is for X0 calcs.

    k = jzlim - idnint( ycur / delz )
    do i = 0, npnts-1
        xp(i) = real( u(k), 8 )
        yp(i) = dimag(u(k))
        k = k - 1
    end do

else

! Store lower NPNTS of U() in XP() and YP().  This is for rough surface calcs.

    xp(0:npnts-1) = real( u(0:npnts-1), 8 )
    yp(0:npnts-1) = dimag( u(0:npnts-1) )

end if

do i = np34, npnts
    attn = filtp(i-np34)
    xp(i)=attn*xp(i)
    yp(i)=attn*yp(i)
end do

! Zero pad.

do i = npnts+1, nsm1
    xp(i) = 0.
    yp(i) = 0.
end do

! Transform to obtain spectral field

call drst( xp, lnp, isn )
call drst( yp, lnp, isn )

! Determine amplitude.

do i = 0, nsm1
    xpi = xp(i)
    ypi = yp(i)
    pp = dsqrt( xpi*xpi + ypi*ypi )
    amp = dmax1(1.d-10, pp)
    spectr(i) = 10.* dlog10(amp)
end do

! Perform a 3-point average and look for amplitude peak.

ipeak = 0
peak = -200.

do i = 1, nsm1

    im1 = max0( 1, i-1 )

```

```

        ip1 = min0( nsm1, i+1 )

        p1 = spectr(im1)
        p = spectr(i)
        p2= spectr(ip1)

        pavg = (p1 + p + p2) / 3.
        if( pavg .gt. peak) then
            ipeak = i
            peak = pavg
        end if
    end do

    ! Determine angle from bin# IPEAK where peak occurs.

    thout = dasin( xocon * dble(ipeak) )

end subroutine specest

```

A.2.19 Subroutine TROPOSCAT

```

! ***** SUBROUTINE TROPOSCAT *****

! Module Name: TROPOSCAT

! Module Security Classification: UNCLASSIFIED

! Purpose: This routine determines the loss due to troposcatter and computes
!          the appropriate loss from troposcatter and diffraction beyond the
!          radio horizon.

! Version Number: 1.3.1

! INPUTS:
!   Argument List: ISTEP, JS, JE
!   Common: EK, FTER, IPE, JT2, R1T, RF, SNREF_TX, THETA1S,
!           TLSTWR, TWOKA
!   Public: ADIF(), D2S(), RDT(), RLOG0(), RLOSS(), RNGOUT(), TH1(),
!           THETA0(), THETA2S(), TYH(), ZOUT()

! OUTPUTS:
!   Argument List: NONE
!   Common: NONE
!   Public: RLOSS()

! Modules Used: APM_MOD

! Calling Routines: CALCLOS, EXT0

! Routines Called:
!   APM Specific: ANTPAT
!   Intrinsic: AMAX0, DBLE, DEXP, DLOG10, DMAX1, DMIN1, MINLOC

! GLOSSARY:

!   Input Variables:
!       ISTEP = Current output range step index.
!       JS = Starting index in ZOUT() for troposcatter calculations.
!       JE = Ending index in ZOUT() for troposcatter calculations.
!       RLOSS() = Propagation loss in dB vs. height at range ROUT.

!   Output Variables
!       RLOSS() = Propagation/troposcatter loss in dB vs. height at range ROUT.

!   Local Variables:
!       A1MIN = 1-element array containing subscript of minimum value
!               in TH() array.
!       AL = Angle defined by equ. 115 in EREPS 3.0 User's Manual
!             NRaD TD 2648, pp. 105.
!       ALD = Log of antenna pattern factor for ALPHAD where ALPHAD here

```

```

!           represents lowest direct ray angle in optical region.
!   BE = Angle defined in equ. 116 in EREPS 3.0 User's Manual
!         NRAd TD 2648, pp. 105.
!   BIGH = Frequency gain function defined in equ. 119 in EREPS 3.0
!         User's Manual NRAd TD 2648, pp. 106.
!   CT1 = Quantity defined in equ. 124 in EREPS 3.0 User's Manual
!         NRAd TD 2648, pp. 106.
!   CT2 = Quantity defined in equ. 125 in EREPS 3.0 User's Manual
!         NRAd TD 2648, pp. 106.
!   D1 = Range from source to tangent point in meters.
!   D2 = Range from receiver to tangent point in meters.
!   DELHO = Frequency gain function correction term defined in equ.
!         127 in EREPS 3.0 User's Manual NRAd TD 2648, pp. 106.
!   ER = Exponent used in troposcatter calcs = -4/3
!   ETAS = Quantity defined in equ. 126 in EREPS 3.0 User's Manual
!         NRAd TD 2648, pp. 106.
!   H0 = Effective scattering height - defined in equ. 109 in
!         EREPS 3.0 User's Manual NRAd TD 2648, pp. 105.
!   HOR1 = Quantity defined in equ. 120 in EREPS 3.0 User's Manual
!         NRAd TD 2648, pp. 106.
!   HOR2 = Quantity defined in equ. 121 in EREPS 3.0 User's Manual
!         NRAd TD 2648, pp. 106.
!   JT1 = Index counter for TH1() array.
!   JZ = Current output height index.
!   QT = Quantity defined in equ. 128 in EREPS 3.0 User's Manual
!         NRAd TD 2648, pp. 107.
!   R1 = Quantity defined in equ. 122 in EREPS 3.0 User's Manual
!         NRAd TD 2648, pp. 106.
!   R2 = Quantity defined in equ. 123 in EREPS 3.0 User's Manual
!         NRAd TD 2648, pp. 106.
!   ROUT = Current output range in meters.
!   ROUT3 = Current output range in km.
!   S = Quantity defined equ. 110 in EREPS 3.0 User's Manual
!         NRAd TD 2648, pp. 105.
!   SN1 = Surface refractivity term.
!   SNREF = Average surface refractivity.
!   SNREF_TR = Surface refractivity at receiver range.
!   THETA = Common volume scattering angle in radians.
!   THETA1 = Tangent angle from source height.
!   THETA2 = Tangent angle from receiver height.
!   TLOSS = Troposcatter loss in dB.
!   TLST = Troposcatter loss term.
!   TLSTS = Troposcatter loss term for smooth surface.

```

```

subroutine troposcat( istp, js, je )

```

```

  use apm_mod

```

```

  implicit integer(kind=4) (i-n)
  implicit real(kind=8) (a-h, o-z)

```

```

  integer(kind=4) a1min(1)
  data er / -1.333333d0 /

```

```

  rout = rngout(istp)

```

```

!Initialize surface refractivity and associated terms.

```

```

  snref_tr = refref(0)
  snref = .5 * (snref_tx + snref_tr)
  sn1 = .031 - .00232 * snref + 5.67d-6 * snref**2
  tlsts = tlstwr - .2 * snref

```

```

!For smooth surface, initialize tangent angle for source height and
!initialize troposcatter loss term, plus other variables dependent only
!on range.

```

```

  theta1 = theta1s
  tlst = tlsts
  theta02 = theta0(istp) * .5
  rout3 = rout * 1.d-3

```

```

!If terrain case, determine tangent angle from source and initialize
!counter for receiver case.

if( fter ) then
  do while( ( rout .gt. dble(jt2)*dr ) .and. ( jt2 .le. ipe ))
    jt2 = jt2 + 1
  end do
  j2m = amax0( 1, jt2 - 1 )
  a1min = minloc( th1(1:j2m) )
  jt1 = a1min(1)
  theta1 = th1(jt1)
  d1 = dble(jt1) * dr
end if

do jz = js, je

!For smooth surface, if current output range is less than minimum
!diffraction field range, then still in interference region - exit.

  if( ( rout .lt. rdt(jz) ) .and. ( .not. fter )) return

!For smooth surface, initialize tangent angle for receiver height.

  theta2 = theta2s(jz)
  d2 = d2s(jz)

  if( fter ) then

!If terrain case, determine tangent angle from receiver.

    do i = j2m, jt1, -1
      h2 = tyh(i)
      rx = dble(i) * dr
      r2 = rout - rx
      ang2 = (zout(jz) - h2) / r2 + r2 / twoka
      if( i .eq. j2m ) then
        theta2 = ang2
        d2 = r2
      end if
      if( ang2 .lt. theta2 ) then
        theta2 = ang2
        d2 = r2
      end if
    end do
    if( theta2 .gt. theta2s(jz)) then
      theta2 = theta2s(jz)
      d2 = d2s(jz)
    end if
    if( rout .lt. (d1+d2) ) return

!Get antenna pattern loss term, ALD, based on tangent angle from
!source over terrain.

    alphad = theta1 + 1.d-6
    call antpat( alphad, FACTR )
    if( factr .ne. 0. ) ald = 20. * dlog10( factr )

!Adjust troposcatter loss term.

    tlst = tlsts - ald
  end if

!Determine common volume scattering angle.

  theta = theta0(istp) - theta1 - theta2

  antdifr = adif(jz) / rout

!Determine angles illustrated and defined in equs. 115 and 116 in
!EREPS 3.0 User's manual.

```

```

    al = theta02 - theta1 + antdifr
    be = theta02 - theta2 - antdifr

    s = dmin1( dmax1( .1, al / be ), 10. )

!Get effective scattering height, H0.

    h0 = s * rout3 * theta / ( 1. + s )**2

!The following variables are determined to compute the frequency gain
!function BIGH. All variables are defined in equs. 119-128 in EREPS
!3.0 user's manual.

    etas = .5696 * h0 * ( 1. + sn1 * dexp(-3.8d-6 * h0**6) )
    etas = dmin1( dmax1( .01, etas ), 5. )

    ct1 = 16.3 + 13.3*etas
    ct2 = .4 + .16*etas

    r1 = dmax1( .1, r1t * theta )
    r2 = dmax1( .1, rf * zout(jz) * theta )
    hor1 = dmax1( 0., ct1 * (r1 + ct2)**er )
    hor2 = dmax1( 0., ct1 * (r2 + ct2)**er )

    qt = dmin1( dmax1( .1, r2 / s / r1 ), 10. )

    delho = 6.*( .6 - dlog10(etas) ) * dlog10(s) * dlog10(qt)

    hp = (hor1 + hor2) / 2.
    delho = dmin1( hp, delho )
    if( delho+hp .lt. 0. ) delho = -hp
    bigh = hp + delho

!Troposcatter loss is computed.

    tloss = tlst + 573. * theta + rlogo(istp) + bigh

!Troposcatter loss is compared to propagation loss. If the difference
!between the propagation and troposcatter loss is less than 18 dB,
!then the the propagation factors are summed. If the difference is
!greater than 18 dB then lesser of the 2 losses is used.

    dif = rloss(jz) - tloss
    if( dif .ge. 18. ) then
        rloss(jz) = tloss
    elseif( dif .ge. -18. ) then
        rloss(jz) = rloss(jz) - 10.*dlog10( 1. + 10.**(.1*dif) )
    end if
end do

end subroutine troposcat

```

A.3 SUBROUTINE XOINT

```

! ***** SUBROUTINE XOINIT *****

! Module Name: XOINIT

! Module Security Classification: UNCLASSIFIED

! Purpose: This routine initializes the range, height and angle arrays
!          in preparation for XOSTEP. It performs 2 passes on a 10-pt
!          smoothing average to smooth the propagation angles.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: IXOSTP, JEND
!   Common: FTER, IZ, IZMAX, ZLIM

```

```

!   Public: FFACZ(,)

!   OUTPUTS:
!   Argument List: JXSTART, IERROR
!   Common: NONE
!   Public: CURANG(), CURHT(), CURNG(), IGRD(), PRFH_XO()

!   Modules Included: APM_MOD

!   Calling Routines: MAIN DRIVER PROGRAM

!   Routines Called:
!   APM Specific: APMCLEAR, MEANFILT
!   Intrinsic: ALLOCATE, ALLOCATED, DEALLOCATE

!   GLOSSARY: See universal glossary for common variables and parameters.

!   Input Variables:
!       JEND = Output index in MPFL() where loss values calculated from
!             PE model ends.

!   Output Variables:
!       JXSTART = Output index in MPFL() where loss values calculated from
!             from XO model begins.

!   Local Variables:
!       DUM = Dummy array for CURANG().

subroutine xoinit( ixostp, jend, JXSTART, IERROR )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

real(kind=8), allocatable :: dum(:)

if( ixostp .gt. 0 ) then

    if( allocated( curang ) ) deallocate( curang, stat=ierror )
    allocate( curang(izmax), stat=ierror )
    if( ierror .ne. 0 ) return
    curang = ffacz(:,3)

    if( allocated( curht ) ) deallocate( curht, stat=ierror )
    allocate( curht(izmax), stat=ierror )
    if( ierror .ne. 0 ) return

!   Initialize so that ray tracing in subroutine EXT0 begins at height
!   ZLIM with the first gradient at index 0 in array GRAD(,).

    curht = zlim

    if( allocated( curng ) ) deallocate( curng, stat=ierror )
    allocate( curng(izmax), stat=ierror )
    if( ierror .ne. 0 ) return
    curng = ffacz(:,2)

    if( allocated( igrd ) ) deallocate( igrd, stat=ierror )
    allocate( igrd(izmax), stat=ierror )
    if( ierror .ne. 0 ) return
    igrd = 0.

    if( allocated( prfh_xo ) ) deallocate( prfh_xo, stat=ierror )
    allocate( prfh_xo(izmax,2), stat=ierror )
    if( ierror .ne. 0 ) return
    prfh_xo = 0.d0

    if( allocated( dum ) ) deallocate( dum, stat=ierror )
    allocate( dum(izmax), stat=ierror )
    if( ierror .ne. 0 ) return

```

```

    dum = 0.d0
    if( fter ) then
! Now perform 1st smoothing on entire angle array.
        call meanfilt( curang, iz, ism, DUM )
! Now perform 2nd smoothing on entire angle array.
        call meanfilt( dum, iz, ism, CURANG )
    end if
    jxstart = jend + 1
    deallocate( dum )
else
! Deallocate all allocated arrays.
    call apmclean( IERROR )
end if
end subroutine xoinit

```

A.3.1 Subroutine APMCLEAN

```

!***** SUBROUTINE APMCLEAN *****
! Module Name: APMCLEAN
! Module Security Classification: UNCLASSIFIED
! Purpose: This routine deallocates all dynamically dimensioned arrays
!          used in one complete run of APM calculations.
! Version Number: 1.3.0
! Modified by S. Fast to not return until all deallocation is complete!
! INPUTS:
!   Argument List: IERROR
!   Common: LN
!   Public: Most dynamically dimensioned arrays in APM_MOD
! OUTPUTS:
!   Argument List: IERROR
!   Common: None
!   Public: None
! Modules Used: APM_MOD
! Calling Routines: XOINIT, XOSTEP
! Routines called:
!   APM Specific: DRST
!   Intrinsic: ALLOCATE, ALLOCATED, DEALLOCATE
! GLOSSARY: See universal glossary for common and public variables.
!   Input Variables: None
!   Output Variables: None
!                   KERROR = Integer variable indicating error # for DEALLOCATE and ALLOCATE
!                   statements.
!   Local Variables: None
!                   IERROR = Integer variable indicating error # for DEALLOCATE and ALLOCATE

```



```

!                                statements.

subroutine apmclean( KERROR )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

ierror = 0
kerror = 0

! Deallocate all arrays allocated in ALLARRAY_APM.

if( allocated( fsl ) ) deallocate( fsl, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( hfangr ) ) deallocate( hfangr, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( rsqrd ) ) deallocate( rsqrd, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( rlogo ) ) deallocate( rlogo, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( rngout ) ) deallocate( rngout, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( zout ) ) deallocate( zout, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( zro ) ) deallocate( zro, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( zoutma ) ) deallocate( zoutma, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( zoutpa ) ) deallocate( zoutpa, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( hlim ) ) deallocate( hlim, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( htfe ) ) deallocate( htfe, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( rfac1 ) ) deallocate( rfac1, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( rfac2 ) ) deallocate( rfac2, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( rloss ) ) deallocate( rloss, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( dielec ) ) deallocate( dielec, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( igrnd ) ) deallocate( igrnd, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( rgrnd ) ) deallocate( rgrnd, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( refdum ) ) deallocate( refdum, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( htdum ) ) deallocate( htdum, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

```

```

if( allocated( grdum ) ) deallocate( grdum, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( href ) ) deallocate( href, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( refref ) ) deallocate( refref, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( gr ) ) deallocate( gr, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( q ) ) deallocate( q, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( rm ) ) deallocate( rm, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( zrt ) ) deallocate( zrt, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

! Deallocate arrays used in troposcatter calculations.

if( allocated( adif ) ) deallocate( adif, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( d2s ) ) deallocate( d2s, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( rdt ) ) deallocate( rdt, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( th1 ) ) deallocate( th1, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( theta0 ) ) deallocate( theta0, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( theta2s ) ) deallocate( theta2s, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

! Deallocate all arrays allocated in ALLARRAY_PE.

call drst( udum, ln, -1 ) !Deallocates arrays in DRST module.

if( allocated( envpr ) ) deallocate( envpr, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( filt ) ) deallocate( filt, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( frsp ) ) deallocate( frsp, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( ht ) ) deallocate( ht, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( profint ) ) deallocate( profint, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( rn ) ) deallocate( rn, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( u ) ) deallocate( u, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( ulst ) ) deallocate( ulst, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( w ) ) deallocate( w, stat=ierror )

```

```

if( ierror .ne. 0 ) kerror = ierror

if( allocated( ym ) ) deallocate( ym, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( udum ) ) deallocate( udum, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( cn2 ) ) deallocate( cn2, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

! Deallocate all arrays allocated in ALLARRAY_XORUF.

if( allocated( ffrout ) ) deallocate( ffrout, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( ffacz ) ) deallocate( ffacz, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( grad ) ) deallocate( grad, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( htr ) ) deallocate( htr, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( lvl ) ) deallocate( lvl, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( filtp ) ) deallocate( filtp, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( xp ) ) deallocate( xp, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( yp ) ) deallocate( yp, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( spectr ) ) deallocate( spectr, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( curang ) ) deallocate( curang, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( curht ) ) deallocate( curht, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( curng ) ) deallocate( curng, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( igrd ) ) deallocate( igrd, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( prfh_xo ) ) deallocate( prfh_xo, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( graze ) ) deallocate( graze, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

if( allocated( tyh ) ) deallocate( tyh, stat=ierror )
if( ierror .ne. 0 ) kerror = ierror

end subroutine apmclean

```

A.3.2 Subroutine MEANFILT

```

! ***** SUBROUTINE MEANFILT *****

! Module Name: MEANFILT

! Module Security Classification: UNCLASSIFIED

```

```

! Purpose: Performs ISZ-pt average smoothing/filtering.
! Version Number: 1.3.0
! INPUTS:
!   Argument List: ARBEF(), ISZ, M
!   Common: NONE
! OUTPUTS:
!   Argument List: ARAFT()
!   Common: NONE
! Modules Used: NONE
! Calling Routines: X0INIT
! Routines called:
!   APM Specific: NONE
!   Intrinsic: REAL, SUM
! GLOSSARY: See universal glossary for common variables.
!   Input Variables:
!     ARBEF = array before smoothing.
!     ISZ = # of points in which to take average smoothing.
!     M = # of points in array.
!   Output Variables:
!     ARAFT = array after smoothing.
subroutine meanfilt( arbef, m, isz, ARAFT )
!M must be odd
implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)
dimension arbef(*), araft(*)
mx = (m - 1) / 2
ind = isz - mx
araft(1:isz) = arbef(1:isz)
do k = mx+1, ind
  araft(k) = sum( arbef(k-mx:k+mx) ) / real(m,8)
end do
end subroutine meanfilt

```

A.4 SUBROUTINE XOSTEP

```

!***** SUBROUTINE XOSTEP *****
! Module Name: XOSTEP
! Module Security Classification: UNCLASSIFIED
! Purpose: Calculates loss values in the height region above
!          the maximum height of the PE model for one range step.
! Version Number: 1.3.0
! INPUTS:
!   Argument List: ISTEP, JXSTART
!   Common: GASATT, HTLIM, IHYBRID, NZOUT
!   Public: HTFE(), RNGOUT(), ZOUT()
!   Data: INVALID
! OUTPUTS:

```

```

!   Argument List: JXEND, MPFL(), ROUT
!   Common: GASLOSS

!   Modules Used: APM_MOD

!   Calling Routines: MAIN DRIVER PROGRAM

!   Routines Called:
!   APM Specific: APMCLEAR, EXT0, FEM, ROLOSS
!   Intrinsic: IIDNNT, MAX0

!   GLOSSARY: See universal glossary for common variables.

!   Input Variables:
!       ISTEP = Current output range step index.
!       JXSTART = Output index in MPFL() where loss values calculated
!                 from FE/RO/XO model begins.

!   Output Variables:
!       JXEND = Index at which the valid propagation loss values end.
!       ROUT = Current output range in meters.
!       MPFL(,) = 2-byte integer array containing propagation factor(F) and loss
!                 values in centibels vs. height, at each output range ROUT.
!                 All values returned are referenced to height HMIN.
!                 MPFL(1,i) = loss at output height equal to i*DZOUT
!                 MPFL(2,i) = 20*log10(F) at output height equal to i*DZOUT

!   Local Variables:
!       JFE = ending index within MPFL() of FE loss values.
!       JFS = starting index within MPFL() of FE loss values.
!       JRE = ending index within MPFL() of RO loss values.
!       JRS = starting index within MPFL() of RO loss values.
!       RSQ = Square of output range ROUT

subroutine xostep( istp, ROUT, MPFL, jxstart, JXEND )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

integer(kind=2) mpfl(2,0:*)

if( ihybrid .eq. 0 ) return

jfs = 0
jfe = 0
jrs = 0
jre = 0

rout = rngout(istp)
gasloss = rout * gasatt

mpfl(:,jxstart:nzout) = inval

! Perform extended optics calculations.
! JXE = ending index within MPFL() of XO loss values.

call exto( istp, rout, MPFL, jxstart, JXE )

if( ihybrid .eq. 1 ) then
    if( htfe(istp) .lt. (htlim - htfe(istp)*1.d-5) ) then
        j = nzout
        do while( zout(j) .gt. htfe(istp) )
            j = j - 1
        end do
        jfs = max0( jxe+1, j+1 )
        jfe = nzout
    end if

    if( jfe .gt. 0 ) call fem( istp, rout, MPFL, jfs, jfe )
end if

```

```

! Perform R0 calculations if necessary
! JRS = starting index within MPFL() of R0 loss values.
! JRE = ending index within MPFL() of R0 loss values.

      if( jxe .lt. nzout ) then
        jre = jfs - 1
        if( jre .lt. 0 ) jre = nzout
        jrs = jxe + 1
        if( jrs .gt. jre ) then
          jrs = 0
          jre = 0
        end if
        if( jre .gt. 0 ) call roloss( istp, rout, jrs, jre, MPFL )
      end if
end if

jxend = max0( jxe, jfe, jre )

! Deallocate all allocated arrays.

if( istp .eq. nrout ) call apmclean( IERROR )

end subroutine xostep

```

A.4.1 Subroutine EXT0

```

! ***** SUBROUTINE EXT0 *****

! Module Name: EXT0

! Module Security Classification: UNCLASSIFIED

! Purpose: This routine calculates loss based on X0 techniques. It
!          performs a ray trace on all rays within one output range step
!          and returns the propagation loss up to the necessary height,
!          storing all angle, height, and range information for ray
!          trace upon next call.

! Version Number: 1.3.0

! INPUTS:
!   Argument List: ISTP, JXS, ROUT
!   Common: GASLOSS, FTER, HTLIM, IRATZ, IZ, NZOUT, TROPO
!   Public: CURANG(), CURHT(), CURNG(), FFACZ(), FFROUT(), FSL(),
!           GRAD(), HLIM(), HTR(), IGRD(), LVL(), ZOUT()

! OUTPUTS:
!   Argument List: JXE, MPFL()
!   Common: CURANG(), CURHT(), CURNG(), HLIM(), PRFH_X0(), RLOSS()

! SAVE: IRPS, IZE, IZS

! Modules Used: APM_MOD

! Calling Routines: XOSTEP

! Routines called:
!   APM Specific: PLINT(function), TROPOSCAT
!   Intrinsic: MAX0, MIN0, DMIN1, DSQRT, IIDNNT

! GLOSSARY: See universal glossary for common variables and parameters.

!   Input Variables:
!     ISTP = index of current output range step.
!     JXS = index in MPFL() where loss values calculated from
!           X0 model begins.
!     ROUT = current output range in meters.

!   Output Variables:

```

```

!       JXE = index in MPFL() where loss values calculated from
!       XO model ends.
!       MPFL(,) = 2-byte integer array containing propagation factor(F) and loss
!       values in centibels vs. height, at each output range ROUT.
!       All values returned are referenced to height HMIN.
!       MPFL(1,i) = loss at output height equal to i*DZOUT
!       MPFL(2,i) = 20*log10(F) at output height equal to i*DZOUT

!   Local Variables:
!       A0 = Angle at start of trace in radians.
!       A1 = Angle at end of trace in radians.
!       FFAC = Propagation factor in dB for specified output height point
!             at range ROUT.
!       GRD = Gradient of current refractivity layer being traced through.
!       H0 = Height at start of trace in meters.
!       H1 = Height at end of trace in meters.
!       IGRAD = Index of current gradient level in GRAD(,) in ray trace.
!       IRP = Counter for current refractivity/gradient profile being used
!            from GRAD(,). (Profile varies only for range-dependent case).
!       IRPS = Starting index counter, used to make sure IRP is
!             initialized properly.
!       IZE = Ending index in CURANG(), CURNG(), and CURHT() to trace to
!             ROUT.
!       IZS = Starting index in CURANG(), CURNG(), and CURHT() to trace to
!             ROUT.
!       NX0 = # of rays traced, i.e., height points, in XO region.
!       P1 = Propagation factor at height Z1.
!       P2 = Propagation factor at height Z2.
!       R0 = Range at start of trace in meters.
!       R1 = Range at end of trace in meters.
!       Z1 = Nearest traced height point below current output height point
!            in ZOUT().
!       Z2 = Nearest traced height point above current output height point
!            in ZOUT().

subroutine exto( istp, rout, MPFL, jxs, JXE )

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

integer(kind=2) mpfl(2,0:*)

save irps, ize, izs

! Define in line ray trace functions:

rada1( a, b ) = a**2 + 2. * grd * b           !a=a0, b=h1-h0
rp( a, b ) = a + b / grd                     !a=r0, b=a1-a0
ap( a, b ) = a + b * grd                     !a=a0, b=r1-r0
hp( a, b, c ) = a + ( b**2 - c**2 ) / 2. / grd !a=h0, b=a1, c=a0

! If this is the first time called, then initialize all index variables.

if( istp .eq. iratz ) then
    ize = 1
    izs = 1
    irps = 1
end if

do j = ize, iz
    if( curng(j) .gt. rout ) exit
end do
l = max0( 1, j-1 )
ize = min0( l, iz )

k = 0

! Begin trace.

```

```

do j = ize, 1
  a0 = curang(j)
  r0 = curng(j)
  h0 = curht(j)
  igrad = igrd(j)

  irp = max0( j, irps )
  grd = grad(igrad,irp)

  do while ( r0 .lt. rout )

    if( irp .eq. iz ) then
      r1 = rout
    else
      r1 = dmin1( ffacz(irp+1,2), rout )
    end if

    a1 = ap( a0, r1-r0 )
    h1 = hp( h0, a1, a0 )

    htrx = htr(igrad+1,irp)
    if( h1 .gt. htrx )then
      h1 = htrx
      rad = rada1( a0, h1-h0 )
      a1 = dsqrt( rad )
      r1 = rp( r0, a1-a0 )
      igrad = min0( igrad+1, lvl(irp)-1 )
    end if

    a0 = a1
    r0 = r1
    h0 = h1

    if( r0 .gt. (ffacz(irp+1,2) - r0*1.e-6) ) irp = min0(irp+1, ize)

  end do

! After trace, all angle, range, and height information are stored for
! next call to EXT0. Propagation factor and height at current range step
! ROUT are stored in PRFH_X0().

  curht(j) = h0
  curng(j) = r0
  curang(j) = a0
  igrd(j) = igrad
  k = k + 1
  prfh_xo(k,1) = ffacz(j,1)
  prfh_xo(k,2) = h0
end do

irps = ize

k = k + 1
prfh_xo(k,1) = ffrou(istp,1)
prfh_xo(k,2) = ffrou(istp,2)
nxo = k

! Adjust counter of first starting point for ray tracing if ray has
! already been traced beyond maximum calculation height.

do while( curht(ize) .gt. htlim )
  ize = ize + 1
end do
ize = max0( 1, ize - 1 )

! Sort height and propagation factor, such that PRFH_X0(:,2) contains steadily
! increasing height from PRFH_X0(NX0,2) to PRFH_X0(1,2).

if( fter ) then
  k = 1
  do while( k .gt. 0 )

```



```

      k = 0
      do j = 1, nxo-1
        if( prfh_xo(j,2) .lt. prfh_xo(j+1,2) ) then
          k = j
          hk = prfh_xo(j,2)
          prfh_xo(j,2) = prfh_xo(j+1,2)
          prfh_xo(j+1,2) = hk
          hk = prfh_xo(j,1)
          prfh_xo(j,1) = prfh_xo(j+1,1)
          prfh_xo(j+1,1) = hk
        end if
      end do
    end do
  end if

  jxe = nzout
  do while( zout(jxe) .gt. prfh_xo(1,2) )
    jxe = jxe - 1
  end do
  hlim(istp) = zout(jxe)
  ix = nxo

  ! Now begin interpolation of propagation factor at specified output
  ! points ZOUT(i).

  z1 = prfh_xo(ix,2)
  z2 = prfh_xo(ix-1,2)
  p1 = prfh_xo(ix,1)
  p2 = prfh_xo(ix-1,1)

  do j = jxs, jxe
    z = zout(j)

    do while(( z .gt. z2 ) .and. ( ix .gt. 1 ))
      ix = ix - 1
      if( ix .gt. 1 ) then
        z1 = z2
        p1 = p2
        z2 = prfh_xo(ix-1,2)
        p2 = prfh_xo(ix-1,1)
      end if
    end do

    frac = ( z - z1 ) / ( z2 - z1 )
    ffac = plint( p1, p2, frac )
    rloss(j) = fsl(istp) - ffac
  end do

  ! Compute troposcatter loss.

  if( tropo ) call troposcat( istp, jxs, jxe )

  ! Include gaseous absorption loss.

  rloss(jxs:jxe) = rloss(jxs:jxe) + gasloss

  ! Store propagation loss and factor in MPFL(,).

  mpfl(1,jxs:jxe) = iidnnt( 10. * rloss(jxs:jxe) )
  mpfl(2,jxs:jxe) = iidnnt( 10. * ( fsl(istp)-rloss(jxs:jxe) ) )

end subroutine exto

```

A.5 MODULE APM_MOD

```

module apm_mod

  real( kind=8 ) pi
  parameter ( pi = 3.1415926535897932d0 )      !Self-explanatory
  parameter ( irtemp = 200 )                  !Number of range steps for use in ray-tracing

```

```

!to determine maximum PE angle.

! ERRORFLAG:
! LERR6 = Logical flag that allows for greater flexibility in allowing error
!         -6 to be bypassed. If set to .TRUE. then trapping for this error
!         occurs, otherwise it can be totally ignored by main driver program.
!         (Within the APM program it is handled as a warning). If this
!         error is bypassed (LERR6 = .FALSE.) terrain profile is extended to
!         RMAX with same elevation height of last valid terrain profile point.
! LERR12 = Same as LERR6 - allows for trapping of this error. If LERR12 =
!         .FALSE., then (for range-dependent case) if range of last
!         refractivity profile entered is less than RMAX, the environment
!         is treated as homogeneous from the last profile entered to RMAX.

common / errorflag / lerr6, lerr12
logical( kind=2 ) lerr6, lerr12

! INPUTVAR:
! HMAX = Maximum output height with respect to m.s.l. in meters. If PEFLAG=.FALSE.
!         then HMAX must be at least 100 m.
! HMIN = Minimum output height with respect to m.s.l. in meters. If PEFLAG=.FALSE.
!         then HMIN must be specified such that HMAX-HMIN is at least 100 m.
! NZOUT = Integer number of output height points desired
! NROUT = Integer number of output range points desired
! PEFLAG = Logical flag indicating if pure PE model will be used.
!         *CAUTION*: Setting this flag overrides most error checks and limits.
!         PEFLAG = .TRUE. -> perform entire run using only PE model
!         PEFLAG = .FALSE. -> use normal hybrid models as appropriate
! RMAX = Maximum output range in meters. If PEFLAG=.FALSE. then RMAX must be at
!         least 5 km.
! RMULT = PE range-step multiplier. Multiplies the internally-computed PE
!         range step by the factor RMULT. This must be specified if
!         PEFLAG=.TRUE. and a greater or lower value for the automatically-
!         calculated range step is desired. Default value is 1.
! THMAX = Maximum calculation PE propagation angle in degrees. This must be
!         specified if PEFLAG=.TRUE. Otherwise this value is ignored.
! TROPO = Logical flag indicating if troposcatter calculations are
!         to be performed:
!         TROPO = .FALSE. -> no troposcatter calculations
!         TROPO = .TRUE. -> perform troposcatter calculations

common / inputvar / hmax, hmin, rmax, rmult, thmax, nzout, nROUT, peflag, tropo

real( kind=8 ) hmax, hmin, rmax, rmult, thmax
integer( kind=4 ) nzout, nROUT
logical( kind=4 ) peflag, tropo

! REFRACTIVITY common block and associated input variables:
! ABSHUM = Absolute humidity near the surface in g/m3.
! GAMMAA = Gaseous absorption in dB/km.
! HMSL(,) = Dynamically allocated 2-dimensional array of size
!           (0:LVLP,NPROF) containing heights in meters with respect
!           to mean sea level of each profile. Array format must be
!           HMSL(I,J) = height of Ith level of Jth profile. J = 1
!           for range-independent cases.

!
! ***** NOTE: *****
! LVLP is the actual # of height levels occupying 0 to LVLP-1
! elements in array HMSL; there will be an extra point
! unused on input.
! *****

! IEXTRA = Extrapolation flag for refractivity profiles entered in combination
!         with terrain below m.s.l.
!         IEXTRA = 0 -> extrapolate to minimum terrain height using
!         standard atmosphere gradient.
!         IEXTRA = 1 -> extrapolate to minimum terrain height using
!         first gradient in profile.
! LVLP = Number of levels in refractivity profile (for range dependent
!         case all profiles must have same number of levels).
! NPROF = Number of profiles - equals 1 for range-independent cases.

```

```

!   NW = Number of wind speeds and corresponding ranges.
!   REFMSL(,) = Dynamically allocated 2-dimensional array of size
!               (0:LVLP,NPROF) containing refractivity with respect to
!               mean sea level of each profile. Array format must be
!               REFMSL(I,J) = M-unit value at Ith level of Jth profile.
!               J = 1 for range-independent cases.
!
!               ***** NOTE: *****
!               LVLP is the actual # of refractivity levels occupying 0 to
!               LVLP-1 elements in array REFMSL; there will be an extra
!               point unused on input.
!               *****
!
!   RNGPROF() = Ranges of each profile in meters, i.e., RNGPROF(I) = range of
!               Ith profile. RNGPROF(1) should always be equal to 0.
!   RNGWIND() = Ranges in meters, for each wind speed specified in WIND;
!               i.e., RNGWIND(1) should always be equal to 0.
!   TAIR = Air temperature near the surface in degrees C.
!   WIND() = Dynamically allocated array containing wind speeds in
!            meters/second. Must also specify corresponding ranges in
!            RNGWIND(). Wind speeds must be no greater than 10 m/s.
!
! ***** NOTE *****
! For Compaq Visual Fortran Ver. 6.6 Fortran 95 compilation using
! dynamically allocated arrays, the following source code MUST be in the
! main driver (calling) program before initialization of the arrays can
! be performed:
!
! IF( ALLOCATED( HMSL ) ) DEALLOCATE( HMSL )
! ALLOCATE( HMSL(0:LVLP, NPROF) )
! HMSL = 0.
!
! IF( ALLOCATED( REFMSL ) ) DEALLOCATE( REFMSL )
! ALLOCATE( REFMSL(0:LVLP, NPROF) )
! REFMSL = 0.
!
! IF( ALLOCATED( RNGPROF ) ) DEALLOCATE( RNGPROF )
! ALLOCATE( RNGPROF(NPROF) )
! RNGPROF = 0.
!
! IF( ALLOCATED( RNGWIND ) ) DEALLOCATE( RNGWIND )
! ALLOCATE( RNGWIND(NW) )
! RNGWIND = 0.
!
! IF( ALLOCATED( WIND ) ) DEALLOCATE( WIND )
! ALLOCATE( WIND(NW) )
! WIND = 0.
!
! Once the above source code has been inserted in the main (calling)
! routine, the arrays can then be initialized with the desired refrac-
! tivity profiles for subsequent use by routines APMINIT and APMSTEP.
! *****
!
common / refractivity / abshum, gammaa, tair, iextra, nw, lvlp, nprof

real( kind=8 ) abshum, gammaa, tair
integer( kind=4 ) iextra, nw, lvlp, nprof

real( kind=8 ), allocatable :: hmsl(:,,:), refmsl(:,,:), rngprof(:), &
                                rngwind(:), wind(:)
public :: hmsl, refmsl, rngprof, rngwind, wind

! SYSTEMVAR:
!   ANTHT = Transmitting antenna height above local ground in meters. Minimum
!           antenna height will vary depending on frequency and beamwidth specified.
!           Absolute minimum is 1.5 m.
!   BWIDTH = Half-power (3 dB) antenna pattern vertical beamwidth in degrees
!            (.5 to 45.)
!   ELEV = Antenna pattern elevation angle in degrees. (-10 to 10)
!   FREQ = Frequency in MHz. 100 MHz <= FREQ <= 20 GHz
!   HFANG() = Dynamically allocated array of power cut-back angles (IPAT=6) or

```

```

!           angles corresponding to antenna pattern factors (IPAT=7). Angles
!           must be specified in degrees. Angles must be specified in
!           increasing order from lowest to highest.
!           This is only used for antenna type 6 and 7.
!   HFFAC() = Dynamically allocated array of power cut-back factors (IPAT=6)
!           or antenna pattern factors (IPAT=7). This is only used for
!           antenna type 6 and 7.
!   IPAT = Integer value indicating type of antenna pattern desired
!           IPAT = 1 -> omni
!           IPAT = 2 -> gaussian
!           IPAT = 3 -> sinc x
!           IPAT = 4 -> csc**2 x
!           IPAT = 5 -> generic height-finder
!           IPAT = 6 -> user-defined height-finder (must specify power
!           reduction factors and corresponding angles in arrays
!           HFFAC(), HFANG() )
!           IPAT = 7 -> user-defined antenna pattern (must specify antenna
!           pattern factors and corresponding angles in arrays
!           HFFAC(), HFANG() )
!   IPOL = integer indicating polarization.
!           IPOL = 0 -> horizontal
!           IPOL = 1 -> vertical
!   NFACS = Number of user-defined cut-back angles and cut-back antenna
!           pattern factors for user-defined height-finder antenna type.

! ***** NOTE *****
! For Compaq Visual Fortran Ver. 6.6 Fortran 95 compilation using
! dynamically allocated arrays, the following source code MUST be in the
! main driver (calling) program before initialization of the arrays can
! be performed:

! IF( ALLOCATED( HFANG ) ) DEALLOCATE( HFANG )
! ALLOCATE( HFANG(NFACS) )
! HFANG = 0.

! IF( ALLOCATED( HFFAC ) ) DEALLOCATE( HFFAC )
! ALLOCATE( HFFAC(NFACS) )
! HFFAC = 0.

! Once the above source code has been inserted in the main (calling)
! routine, the arrays can then be initialized with the desired cut-back
! angles and factors for subsequent use by routines APMINIT and APMSTEP.
! *****

common / systemvar / anht, bwidth, elev, freq, ipat, ipol, nfacs

real( kind=8 ) anht, bwidth, elev, freq
integer( kind=4 ) ipat, ipol, nfacs

real( kind=8 ), allocatable :: hfang(:), hffac(:)
public :: hfang, hffac

! TERRAIN common block and associated input variables:
! DIELEC(,) = Dynamically allocated 2-dimensional array of size (2,IGR)
!           containing the relative permittivity and conductivity;
!           DIELEC(1,i) and DIELEC(2,i), respectively. Only needs to be
!           specified if using IGRND(i) = 7, otherwise, APM will
!           calculate based on frequency and ground types 0-6.
!   IGR = number of different ground types specified
!   IGRND() = Dynamically allocated integer array of size IGR containing
!           ground type composition for given terrain profile - can
!           vary with range. Different ground types are:
!           0 = sea water
!           1 = fresh water
!           2 = wet ground
!           3 = medium dry ground
!           4 = very dry ground
!           5 = ice at -1 degree C
!           6 = ice at -10 degree C
!           7 = user defined (in which case, values of relative
!           permittivity and conductivity in S/m must be given).

```

```

! ITP = number of height/range pairs in profile
! RGRND() = Dynamically allocated array of size (IGR) containing ranges,
!           in m, at which the ground types apply.
! TERX() = Dynamically allocated array of size (ITP) containing range
!           points of terrain profile in meters.
! TERY() = Dynamically allocated array of size (ITP) containing height
!           points of terrain profile in meters.

! ***** NOTE *****
! For Compaq Visual Fortran Ver. 6.6 Fortran 95 compilation using
! dynamically allocated arrays, the following source code MUST be in the
! main driver (calling) program before initialization of the arrays can
! be performed:

! IF( ALLOCATED( DIELEC ) ) DEALLOCATE( DIELEC )
! ALLOCATE( DIELEC(2, IGR) )
! DIELEC = 0.

! IF( ALLOCATED( IGRND ) ) DEALLOCATE( IGRND )
! ALLOCATE( IGRND(IGR) )
! IGRND = 0.

! IF( ALLOCATED( RGRND ) ) DEALLOCATE( RGRND )
! ALLOCATE( RGRND(IGR) )
! RGRND = 0.

! IF( ALLOCATED( TERX ) ) DEALLOCATE( TERX )
! ALLOCATE( TERX(ITP) )
! TERX = 0.

! IF( ALLOCATED( TERY ) ) DEALLOCATE( TERY )
! ALLOCATE( TERY(ITP) )
! TERY = 0.

! Once the above source code has been inserted in the main (calling)
! routine, the arrays can then be initialized with the desired terrain
! information for subsequent use by routines APMINIT and APMSTEP.
! *****

common / terrain / igr, itp

integer( kind=4 ) igr, itp

real(kind=8), allocatable :: dielec(:, :), rgrnd(:), terx(:), tery(:)
integer, allocatable :: igrnd(:)
public :: dielec, igrnd, rgrnd, terx, tery

! ***** START OF INTERNAL APM DECLARATIONS *****

! Common Block APM_VAR

! AATZ = Local propagation angle in radians at height ZLIM and range RATZ
!       (used for hybrid modes).
! ACRIT = Critical angle (angle above which no rays are trapped)
!         in radians.
! ACUT = Tangent angle in radians from antenna height to horizon range.
! AEK = Mean earth radius times the effective earth radius factor EK.
! AFAC = Constant used in determining antenna pattern factors
!       AFAC = 1.39157 / sin( bw / 2 ) for SIN(X)/X and height-finder
!       AFAC = (.5*ln(2))/(sin(bw/2))**2 for GAUSSIAN
! ALAUNCH = Launch angle used, in radians, which, when traced, separates
!           PE & XO regions from RO region
! ALPHAD = Direct ray elevation angle in radians
! ALPHAQ = Complex surface impedance.
! ANTREF = Transmitting antenna height relative to the reference
!         height HMINTER.
! BW = Antenna pattern beamwidth in radians.
! C1X = Constant dependent on each new calculated RT - used to
!       calculate CK1 at next range step.
! C2X = Constant dependent on each new calculated RT - used to
!       calculate CK2 at next range step.

```

```

! CK1 = Coefficients used in old central difference DMFT calculations.
! CK2 = Coefficients used in old central difference DMFT calculations.
! CMFT = Coefficient used in backward difference DMFT calculations.
! CMFT_X = Constant dependent on each new calculated RT - used to
!         calculate CMFT at next range step.
! CON = 1.e-6 * FKO; Constant used in calculation of ENVPR().
! DELP = Mesh size in angle- (or p-) space.
! DELXRO = R0 range increment in meters
! DELZ = Bin width in z-space = WL / (2*sin(THETAMAX))
! DMAGSQ(,) = direct-ray magnitude squared - used in R0 calculations.
! DR = PE range step in meters
! DR2 = 1/2 PE range step in meters
! DRGRZ = Range step in meters used in estimating grazing angles.
! DROUT = Output range step in meters
! DTHETA = Angle bin width (i.e., incremental sine(theta))
! DZ2 = 2. * DELZ
! DZOUT = Output height increment in meters
! EK = Effective earth radius factor.
! ELV = Antenna pattern elevation angle in radians
! FKO = Free-space wavenumber = (2*pi) / WL
! FK02 = 2. * FKO
! FNORM = Normalization factor used for DFT.
! FTER = logical flag - .TRUE. = terrain case, .FALSE. = over-water case
! GASATT = Gaseous absorption in dB/m.
! GASLOSS = Gaseous absorption loss in dB.
! HMINTER = Minimum height of terrain profile in meters. This will be
!         used to adjust entire terrain profile so all internal
!         calculations will be referenced to this height.
! HMREF = Height relative to HMINTER. Determined from user-provided
!         minimum height HMIN. That is, if HMIN is minimum height input
!         by user with respect to mean sea level, and HMINTER is
!         internally considered the new origin, then HMREF = HMIN - HMINTER.
! HTEMP() = Heights at which ray is traced to every range point RTEMP(i)
! HTLIM = User-supplied maximum height relative to HMINTER, i.e.,
!         HTLIM = HMAX - HMINTER
! HTYDIF = Height difference between internal maximum height, HTLIM,
!         and initial ground height at the source, YFREF.
! IALG = Index indicating which DMFT algorithm is being used:
!         0 = no DMFT being used
!         1 = old central difference algorithm
!         2 = backward difference algorithm
! IAP = Index indicating when local ray angle becomes positive in array
!       RAYA().
! IG = Counter indicating current ground type being modeled.
! IGPE = Number of grazing angles computed from spectral estimation.
! IGRZ = Number of grazing angles computed from ray trace and total # of
!       grazing angles.
! IHMX = Output range step index where maximum height HTLIM is reached in
!       array HLIM().
! IHYBRID = Integer indicating which sub-models will be used:
!           0 = airborne hybrid (FE+PE) model
!           1 = full hybrid (FE+R0+PE+X0) model
!           2 = PE + X0 model
! IO = Starting index for MPFL array:
!       IO=0 -> 1st calculated point is at surface (0.0 height)
!       IO=1 -> 1st calculated point is at height DZOUT
! IPE = Number of PE range steps.
! IRATZ = Index of output range step at which ZLIM is reached (for
!       hybrid model only). Indicates at what range step begin
!       storing propagation factor and outgoing angle for X0 region.
! IRON = Next index for R0 solution (0 or 1)
! IROP = Previous index for R0 solution (-1, 0, or 1)
! IS = Counter for current profile (for range-dependent cases)
! ISM2 = 1/2 of data variable ISM
! ISTART= R0 height index at transmitter.
! ISTART1 = Refractivity level index within HTDUM() at transmitter.
! ITPA = Number of terrain points used internally in arrays TX() and TY().
! IXO = Number of range steps in X0 calculation region.
! IZ = Counter for points stored in FFACZ(,) array.
! IZG = Output height integer index indicating the last height representing
!       terrain.

```

```

! IZINC = Integer increment for storing points at top of PE region to
! start X0 model. I.e., points are stored at every IZINC range
! step.
! IZMAX = Maximum # of points allocated for arrays associated with X0 calcs.
! JT2 = Index counter for TYH() array indicating where range is at each terrain
! inflection, i.e., "range at TYH(JT2-1)" < ROUT < "range at TYH(JT2)".
! JZLIM = PE bin # corresponding to ZLIM, i.e., ZLIM = JZLIM*DELZ.
! KMAX = Maximum K-index at XROP and XRON
! KMINN = Minimum K-index at XRON
! KMINP = Minimum K-index at XROP
! KTR1 = Number of increasing tangent angles and ranges determined from
! source height over terrain path profile.
! LDUCT = Logical flag indicating if surface-based duct profile has been specified:
! .FALSE. = no duct
! .TRUE. = duct
! LEVAP = Logical flag indicating if evaporation duct refractivity profile has been
! specified:
! .FALSE. = no evaporation duct
! .TRUE. = evaporation duct
! LEVELS = number of levels defined in ZRT(), Q(), and GR() arrays
! LN = Power of 2 transform size, i.e. N = 2**LN
! LNMIN = Minimum power of 2 transform size.
! LNP = Power of 2 transform size used in spectral estimation calcs.
! LVLEP = Number of height/refractivity levels in profile REFDUM(),HTDUM()
! taken w.r.t. reference height HMINTER.
! N = Transform size
! N34 = 3/4 * N
! NF4 = N / 4
! NLVL = Number of height/refractivity levels in profile REFREF(),HREF()
! taken w.r.t. local ground height at middle of range step, YCURM.
! NM1 = N-1
! NOPE = Integer flag indicating if PE calcs are needed:
! 0 = PE calcs are needed.
! 1 = PE calcs are not needed.
! NP34 = 3/4 * NPNTS
! NP4 = NPNTS / 4
! NPNTS = Number of points used in top part of PE region for spectral
! estimation.
! NS = Transform size used in spectral estimation calcs = 2**LNP
! NSM1 = NS - 1
! OMEGA(,) = Phase angle between direct & reflected rays in radians - used in RO
! calculations.
! PELEV = Sine of elevation angle
! PLCNST = Constant used in determining propagation loss
! PLCNST = 20log(2*FKO).
! PSILIM = Grazing angle of limiting ray in radians
! QIFKO = Imaginary i * FKO
! R1T = Constant used in troposcatter calcs. = RF*ANTREF
! RATZ = Range at which ZLIM is reached (used for hybrid model).
! RAYA() = Array containing all local angles of traced ray ALAUNCH at
! each output range.
! RF = Constant used in troposcatter calcs. = 4*PI*FREQ/speed of light (x10e6)
! RHOR = Radar horizon range in meters for 0 receiver height.
! RK = Constant used in central difference DMFT.
! RLOG = 10. * alog10( PE range )
! RLOGLST = RLOG of previous range step (i.e., 10*alog10(PE range-DR) )
! RMAGSQ(,) = Reflected-ray magnitude squared - used in RO calculations
! RMTX = M-unit value * 1.e-6 at transmitter
! RPEST = Range in meters at which loss values from the PE model will
! start being calculated.
! RT = Complex root for mixed transform method based on Kuttler's formulation.
! RTEMP() = Range steps for tracing to determine maximum PE angle.
! RUF = Logical flag indicating if performing a rough sea surface
! calculation:
! .FALSE. = smooth sea surface; 0 wind speed
! .TRUE. = rough sea surface; use wind speeds specified in array WIND()
! RUF_FAC = Factor used for wave height calculation.
! RUF_HT = Sea surface wave height.
! RV2 = Range of the next refractivity profile (for range-dependent cases)
! SBW = Sine of the beamwidth.
! SNREF_TX = Surface refractivity at source.

```

```

!   TERANG = Current maximum angle made by tangent line from transmitting
!           antenna height to all terrain elevation points up to the
!           current output range. (Used only for IHYBRID=0)
!   THETA1S = Tangent angle from source for smooth surface.
!   THETA75 = 75% of maximum propagation angle in PE calculations.
!   TLSTWR = Troposcatter loss term for smooth surface (non-terrain), before
!           addition of surface refractivity term.
!   TWOKA = Twice the effective earth's radius factor times the effective
!           earth radius. The effective earth's radius factor is calcula-
!           ted based on a ray trace at 5 degrees from the origin to HTLIM
!           for IHYBRID=1 and from the antenna height to HTLIM for IHYBRID
!           =0. This is used for routine FEM.
!   TWOKA_DOWN = Twice the effective earth's radius factor times the
!           effective earth radius. The effective earth's radius factor
!           is calculated based on a ray trace at -5 degrees from the
!           antenna to the surface. This is used for routine AIRBORNE.
!   UMAX = limiting angle used in cut-off point for SIN(X)/X and
!           generic height-finder antenna pattern factors
!   WL = Wavelength in meters.
!   XLIMRO = range of limiting ray in meters
!   XOCON = Constant used in determining outgoing propagation angle
!           for XO calcs -> WL / (2*NS*DELZ).
!   XRON = next range for R0 solution in meters
!   XROP = previous range for R0 solution in meters
!   YCUR = height of ground at the current range step
!   YCURM = height of ground midway between last and current range step.
!           For use when shifting profiles to be relative to the local ground
!           height.
!   YFREF = Ground elevation height at source.
!   YLAST = height of ground at the last range step
!   ZLIM = Maximum internal height (HTLIM) or .75*ZMAX, whichever is smaller.
!   ZMAX = Maximum height of PE calculation domain = N * DELZ
!   ZTOL = height tolerance for Newton's method in meters

common / apm_var / htemp(irtemp), raya(irtemp), rtemp(irtemp), dmagsq(0:1,0:88), &
             omega(0:1,0:88), rmagsq(0:1,0:88), alphaq, c1x, c2x, ck1, ck2, &
             cmft, cmft_x, qifko, rk, rt, aatz, acrit, acut, aek, afac, &
             alaunch, alimv, alphas, antref, bw, con, delp, delxRO, delz, &
             dr, dr2, drgrz, drout, dtheta, dz2, dzout, ek, elv, fko, fko2, &
             fnorm, gasatt, gasloss, hminter, hmref, htlim, htydif, pelev, &
             pi2, plcns, psilim, r1t, ratz, rf, rhor, rlog, rloglst, rmtx, &
             rpest, ruf_fac, ruf_ht, rv2, sbw, snref_tx, terang, theta1s, &
             theta75, tlstwr, twoka, twoka_down, umax, umax3, wl, xlimRO, &
             xocon, xRON, xROP, ycur, ycurm, yfref, ylast, zlim, zmax, ztol, &
             ialg, iap, ig, igpe, igrz, ihm, ihybrid, io, ipe, iratz, iRON, &
             iROP, is, ism2, istart, istart1, itpa, ixo, iz, izg, izinc, &
             izmax, jt2, jzlim, kmax, kminn, kminp, ktr1, levels, ln, lnmin, &
             lnp, lvlep, n, n34, nf4, nlvl, nm1, nope, np34, np4, npnts, ns, &
             nsm1, fter, lduct, levap, ruf

real( kind=8 ) htemp, raya, rtemp, dmagsq, omega, rmagsq, aatz, acrit, acut, aek, &
             afac, alaunch, alimv, alphas, antref, bw, con, delp, delxRO, delz, &
             dr, dr2, drgrz, drout, dtheta, dz2, dzout, ek, elv, fko, fko2, &
             fnorm, gasatt, gasloss, hminter, hmref, htlim, htydif, pelev, pi2, &
             plcns, psilim, r1t, ratz, rf, rhor, rlog, rloglst, rmtx, rpest, &
             ruf_fac, ruf_ht, rv2, sbw, snref_tx, terang, theta1s, theta75, &
             tlstwr, twoka, twoka_down, umax, umax3, wl, xlimRO, xocon, xRON, &
             xROP, ycur, ycurm, yfref, ylast, zlim, zmax, ztol

integer( kind=4 ) ialg, iap, ig, igpe, igrz, ihm, ihybrid, io, ipe, iratz, iRON, &
             iROP, is, ism2, istart, istart1, itpa, ixo, iz, izg, izinc, &
             izmax, jt2, jzlim, kmax, kminn, kminp, ktr1, levels, ln, &
             lnmin, lnp, lvlep, n, n34, nf4, nlvl, nm1, nope, np34, np4, &
             npnts, ns, nsm1

complex( kind=8 ) alphaq, c1x, c2x, ck1, ck2, cmft, cmft_x, qifko, rk, rt

logical( kind=4 ) fter, lduct, levap, ruf

! ***** DYNAMICALLY ALLOCATED ARRAYS *****

```



```

! ADIF() = Height array in meters used for troposcatter calcs.
! CN2() = Complex dielectric constant.
! CURANG() = Current local angle for each ray being traced in XO region.
! CURHT() = Current local height for each ray being traced in XO region.
! CURNG() = Current local range for each ray being traced in XO region.
! D2S() = Tangent range array in meters for all output receiver heights
!         over smooth surface.
! ENVPR() = Complex array containing refractivity exponential term.
!           i.e. ENVPR() = exp[i * DR * FKO * 1e-6 * M(z) ], where
!           M(z) is the refractivity at each PE bin height z.
! FFACZ(,) = 2-dimensional array containing propagation factor in dB,
!           range, and propagation angle at ZLIM. Used to start XO
!           calculations.
!           FFACZ(IZ,1) = propagation factor in dB at current PE range
!           FFACZ(IZ,2) = current PE range
!           FFACZ(IZ,3) = propagation angle at current PE range at ZLIM
! FFROUT() = Propagation factor in dB at each output range step
!           beyond RATZ and at height ZLIM
! FILT() = Cosine-tapered (Tukey) filter array.
! FILTP() = Array filter for spectral estimation calcs.
! FRSP() = Complex array containing wide-angle free-space propagator exponential
!         term; i.e., FRSP() = exp[-i * DR * (FKO - sqrt(FKO**2 - p**2)) ]
! FSL() = Array containing the free space loss at every output range.
! GR() = 1.E-6 * dM/dz array used for RO calculations
! GRAD(,) = 2-dimensional array containing gradients of each refrac-
!         tivity profile vs. range from height ZLIM to HTLIM.
! GRAZE() = Array of final interpolated grazing angles determined for rough
!         surface calculations.
! GRDUM() = Array of refractivity gradients defined by profile HTDUM(), REFDUM()
! GRZ_PE() = Grazing angles computed from spectral estimation.
! GRZ_RAY() = Grazing angles computed from ray trace
! HFANGR() = Array of user-defined cut-back angles (IPAT=6) or antenna pattern
!         factor angles (IPAT=7) in radians. This is
!         used only for antenna types 6 and 7.
! HLIM() = Array containing height at each output range separating the
!         RO region from the PE (at close ranges) and XO (at farther
!         ranges) regions
! HREF() = Heights of refractivity profile with respect to YREF (local
!         ground height).
! HT() = Height array of size N. Heights space every DELZ.
! HTDUM() = Height array containing height values for current (interpolated)
!         profile in meters, relative to HMINTER.
! HTFE() = Array containing the height at each output range step separa-
!         ting the flat earth region from the RO region.
! HTR(,) = 2-dimensional array containing height levels of each refrac-
!         tivity profile vs. range from height ZLIM to HTLIM.
! IGRD() = Integer indexes indicating at what gradient in GRAD(,) to
!         begin raytracing for next XO range step for each ray in XO
!         region.
! LVL() = Number of refractivity levels in current refractivity profile
!         from ZLIM to HTLIM.
! PRFH_XO(,) = Propagation factor and height for each ray traced in XO region
!         at range ROUT.
! PROFINT() = M-unit profile interpolated to every DELZ in height
! Q() = 2 * [RM(i+1)-RM(i)] array used for RO calculations
! RDT() = Minimum range array (in meters) at which diffraction field
!         solutions are applicable and intermediate region ends (for
!         smooth surface) for all output receiver heights.
! REFDUM() = Dummy array containing M-unit values for current (interpolated)
!         profile taken relative to HMINTER.
! REFREF() = Refractivity array w.r.t. YREF (local ground height).
! RFAC1() = Propagation factor at valid output height points computed
!         from PE field at previous PE range, i.e., ULST().
! RFAC2() = Propagation factor at valid output height points computed
!         from PE field at current PE range, i.e., U().
! RLOGO() = Array of logarithm of output ranges, i.e., RLOGO(i) =
!         20. * ALOG10(i*ROUT).
! RLOSS() = Propagation loss in dB.
! RM() = 1.E-6 * M array used for RO calculations.
! RN() = Array of RT to the i'th power, i.e. RN(I) = RT**I

```

```

!   RNGOUT() = Array containing all output ranges in meters.
!   RSQRD() = Double precision array containing the square of output ranges
!   SLP() = Slope of each segment of terrain.
!   SPECTR() = Field amplitude of spectral portion of PE field in dB.
!   TH1() = Tangent angles from source height w/ terrain path profile.
!   THETA0() = Angle array - angles used in determining common volume
!             scattering angle.
!   THETA2S() = Tangent angle array from all output receiver heights for
!             smooth surface.
!   TX() = Range points of terrain profile in meters.
!   TY() = Adjusted height points of terrain profile in meters.
!   TYH() = Adjusted height points of terrain profile at every PE range step.
!   U() = Complex array containing PE field solution.
!   UDUM() = Dummy array used for temporary storage of real or imaginary part
!           of complex PE field array U().
!   ULST() = Complex array containg PE field solution at previous range step.
!   W() = Difference equation of complex PE field array. Used in
!        intermediate calculations only for vertical polarization.
!   XP() = Real part of spectral portion of PE field.
!   YM() = Field from recursion equation for central difference DMFT.
!   YP() = Imaginary part of spectral portion of PE field.
!   ZOUT() = Array containing all output heights in meters referenced to
!           HMINTER.
!   ZOUTMA() = Array containing output heights in meters relative to the
!             antenna height above ground at 0 range. Used in FE model.
!   ZOUTPA() = Array containing output heights in meters relative to the
!             image antenna height below ground at 0 range. Used in FE
!             model.
!   ZRO() = Output height array in meters referenced to ground elevation
!           height at source. Used for R0 calculations.
!   ZRT() = Height array used for R0 calculations in meters

complex(kind=8), allocatable :: cn2(:,), envpr(:,), frsp(:,), rn(:,), u(:,)    &
                             ulst(:,), w(:,), ym(:,)
public :: cn2, envpr, frsp, rn, u, ulst, w, ym

integer(kind=4), allocatable :: igrd(:,), lvl(:,)
public :: igrd, lvl

real(kind=8), allocatable :: adif(:,), curang(:,), curht(:,), curng(:,), d2s(:,) &
                             ffacz(:,), ffrou(:,), filt(:,), filtp(:,), fsl(:,) &
                             gr(:,), grad(:,), graze(:,), grdum(:,), grz_pe(:,) &
                             grz_ray(:,), hfangr(:,), hlim(:,), href(:,), ht(:,) &
                             htum(:,), htfe(:,), htr(:,), prfh_xo(:,), &
                             profint(:,), q(:,), rdt(:,), refdum(:,), refref(:,) &
                             rfac1(:,), rfac2(:,), rlogo(:,), rloss(:,), rm(:,) &
                             rngout(:,), rsqrd(:,), slp(:,), spectr(:,), th1(:,) &
                             theta0(:,), theta2s(:,), tx(:,), ty(:,), tyh(:,), udum(:,) &
                             xp(:,), yp(:,), zout(:,), zoutma(:,), zoutpa(:,), zro(:,) &
                             zrt(:,)

public :: adif, curang, curht, curng, d2s, ffacz, ffrou, filt, filtp, fsl, &
         gr, grad, graze, grdum, grz_pe, grz_ray, hfangr, hlim, href, ht, &
         htum, htfe, htr, prfh_xo, profint, q, rdt, refdum, refref, rfac1, &
         rfac2, rlogo, rloss, rm, rngout, rsqrd, slp, spectr, th1, theta0, &
         theta2s, tx, ty, tyh, udum, xp, yp, zout, zoutma, zoutpa, zro, zrt

!Data constants

integer( kind=4 ) icn, ism, isn
integer( kind=2 ) interrain, inval
real( kind=8 ) aekst, c0, deg5, deg10, radc, rtst, tan5
complex( kind=8 ) qi

data aekst / 8.4946667d6 /      !4/3 times mean earth radius in m
data c0 / 299.79245 /          !speed of light x 1e-6 m/s
data deg5 / 8.72664626d-2 /    !5. degrees in radians
data deg10 / .17453292d0 /      !10. degrees in radians
data icn, isn / 0, 1 /         !Integer flag indicating type of transform:
                                ! ICN = 0, Cosine transform
                                ! ISN = 1, Sine transform

```

```

data interrains / -999 /      !Integer value used to fill output field value
                                !   array MPFL() indicating output height point
                                !   lies below terrain surface.
data inval / -1000 /         !Integer value used to fill output field value
                                !   array MPFL() indicating invalid propagation loss
                                !   and factor values.
data ism / 9 /               !# of points to smooth propagation angle
                                !   for preparation of XOSTEP routine.
data qi / (0.d0, 1.d0) /     !Imaginary i
data radc / 1.74533d-2 /     !degree to radian conversion factor
data rtst / 2499. /          !Range set at 2.5 km to begin calculation
                                !   of R0 values.
data tan5 / 8.748866353d-2 / !Tangent of 5 degrees

```

SOFTWARE TEST DESCRIPTION
FOR THE
ADVANCED PROPAGATION MODEL CSCI
(Version 1.3.1)

9 August 2002

Prepared for:

Space and Naval Warfare Systems Command (PMW-155)
San Diego, CA

Prepared by:

Space and Naval Warfare Systems Center, San Diego
Atmospheric Propagation Branch (Code 2858)
49170 Propagation Path
San Diego, CA 92152-7385

CONTENTS

1. SCOPE	1
1.1 Identification	1
1.2 Document Overview	1
2. REFERENCE DOCUMENTS	1
3. TEST PREPARATIONS	2
3.1 Hardware Preparation	2
3.2 Software Preparation	2
3.3 Other Pretest Preparation	2
4. TEST DESCRIPTIONS	3
4.1 Requirements Addressed	4
4.2 Prerequisite Conditions	4
4.3 Test Inputs	4
4.4 Expected Test Results	18
4.5 Criteria for Evaluating Results	36
4.6 Test Procedure	36
4.7 Assumptions and Constraints	36
5. REQUIREMENTS TRACEABILITY	36
6. NOTES	37
7. SAMPLE PROGRAM LISTING	39
8. INPUT FILE LISTINGS FOR TEST CASES	46
8.1 ABSORB.IN	46
8.2 AIRBORNE.IN	46
8.3 BLOCK.IN	47
8.4 COSEC2.IN	47
8.5 EDUCT.IN	48
8.6 EDUCTRF.IN	49
8.7 FLTA50.IN	49
8.8 GASABS.IN	50
8.9 GAUSS.IN	50
8.10 HIBW.IN	51
8.11 HIEL.IN	51
8.12 HIFREQ.IN	52
8.13 HITRAN.IN	52
8.14 HORZ.IN	53
8.15 HTFIND.IN	53
8.16 LOBW.IN	54
8.17 LOEL.IN	54
8.18 LOFREQ.IN	55
8.19 LOTRAN.IN	55
8.20 MPRT.IN	56
8.21 PERW.IN	56
8.22 PVT.IN	57
8.23 RDLONGB.IN	58
8.24 RNGDEP.IN	61
8.25 SBDUCT.IN	61
8.26 SBDUCTRF.IN	62
8.27 SINEX.IN	62

8.28 TROPOS.IN	63
8.29 TROPOT.IN	63
8.30 USERDEFA	66
8.31 USERHF.IN	67
8.32 VERT.IN	68
8.33 VERTMIX.IN	68
8.34 VERTSEA.IN	69
8.35 VERTUSRD.IN	69
8.36 WEDGE.IN	70

TABLES

Table 1. Test names and descriptions.	3
Table 2. External environmental data element requirements ^a	5
Table 3. Standard atmosphere with 118-M/km gradient.	7
Table 4. 300-meter surface based duct atmosphere.	7
Table 5. Atmosphere with 14-meter evaporation duct.	7
Table 6. Range-dependent atmosphere, standard atmosphere to surface-based duct.	8
Table 7. Range-dependent atmosphere, surface-based duct to high elevated duct.	8
Table 8. Elevated duct.	8
Table 9. External EM system data element requirements.	9
Table 10. Height-finder angles and factors for case USERDEFA.	10
Table 11. Height-finder angles and factors for case USERHF.	11
Table 12. External implementation data element requirements ^a	12
Table 13. External terrain data element requirements.	13
Table 14. Terrain profile for test case BLOCK.	14
Table 15. Terrain profile for test case FLTA50.	14
Table 16. Terrain profile for test case MPRT.	14
Table 17. Terrain profile for test case PERW.	14
Table 18. Terrain profile for test case PVT.	15
Table 19. Terrain profile for test case RDLONGB and TROPOT.	15
Table 20. Table of ground constants for terrain profile of Table 19.	17
Table 21. Terrain profile for test case VERTMIX.	17
Table 22. Terrain profile for test case VERTUSRD.	17
Table 23. Terrain profile for test case WEDGE.	17
Table 24. Expected output for ABSORB.	18
Table 25. Expected output for AIRBORNE.	18
Table 26. Expected output for BLOCK.	19
Table 27. Expected Output for COSEC2.	19
Table 28. Expected Output for EDUCT.	20
Table 29. Expected output for EDUCTRF.	20
Table 30. Expected output for FLTA50.	21
Table 31. Expected output for GASABS.	21
Table 32. Expected output for GAUSS.	22
Table 33. Expected output for HIBW.	22
Table 34. Expected output for HIEL.	23
Table 35. Expected output for HIFREQ.	23
Table 36. Expected output for HITRAN.	24
Table 37. Expected output for HORZ.	24

Table 38. Expected output for HTFIND.....	25
Table 39 Expected output for LOBW.....	25
Table 40. Expected output for LOEL.....	26
Table 41. Expected output for LOFREQ.....	26
Table 42. Expected output for LOTRAN.....	27
Table 43. Expected output for MPRT.....	27
Table 44. Expected output for PERW.....	28
Table 45. Expected output for PVT.....	28
Table 46. Expected output for RDLONGB.....	29
Table 47. Expected output for RNGDEP.....	29
Table 48. Expected output for SBDUCT.....	30
Table 49. Expected output for SBDUCTRF.....	30
Table 50. Expected output for SINEX.....	31
Table 51. Expected output for TROPOS.....	31
Table 52. Expected output for TROPOT.....	32
Table 53. Expected output for USERDEFA.....	32
Table 54. Expected output for USERHF.....	33
Table 55. Expected output for VERT.....	33
Table 56. Expected output for VERTMIX.....	34
Table 57. Expected output for VERTSEA.....	34
Table 58. Expected output for VERTUSRD.....	35
Table 59. Expected output for WEDGE.....	35
Table 60. Acronyms and abbreviations.....	37

1. SCOPE

1.1 IDENTIFICATION

The Advanced Propagation Model (APM) Version 1.3.1 computer software configuration item (CSCI) calculates range-dependent electromagnetic (EM) system propagation loss and propagation factor within a heterogeneous atmospheric medium over variable terrain, where the radio-frequency index of refraction is allowed to vary both vertically and horizontally. Numerous Naval Integrated Tactical Environmental Subsystem (NITES) applications require EM-system propagation loss values. The APM model described by this document may be applied to two such NITES applications, one which displays propagation loss on a range versus height scale (commonly referred to as a coverage diagram) and one which displays propagation loss on a propagation loss versus range/height scale (commonly referred to as a loss diagram).

1.2 DOCUMENT OVERVIEW

This document specifies the test cases and test procedures necessary to perform qualification testing of the APM CSCI. A discussion of precise input values of each input variable required to perform the test together with final expected test results is presented.

2. REFERENCE DOCUMENTS

1. Commander-In-Chief, Pacific Fleet Meteorological Requirement (PAC MET) 87-04, "Range Dependent Electromagnetic Propagation Models."
2. Naval Oceanographic Office, "Software Documentation Standards and Coding Requirements for Environmental System Product Development," April 1990.
3. Naval Command, Control and Ocean Surveillance Center; Research, Development, Test and Evaluation Division (NRaD), "Terrain Parabolic Equation Model (TPEM) Computer Software Configuration Item (CSCI) Documents," TD 2963, May 1997.
4. Naval Command, Control and Ocean Surveillance Center; Research, Development, Test and Evaluation Division (NRaD), "Radio Physical Optics (RPO) CSCI Software Documents, RPO Ver. 1.16," TD 2403 Rev. 1, April 1997.
5. Space and Naval Warfare Systems Center San Diego (SSC San Diego), "Software Requirements Specification for the Advanced Propagation Model (APM) CSCI (Version 1.3.1)," in TD 3145. SSC San Diego, San Diego, CA. August 2002.

6. Space and Naval Warfare Systems Center San Diego (SSC SD), “Software Design Document for the Advanced Propagation Model (APM) CSCI (Version 1.3.1),” in TD 3145. SSC San Diego, San Diego, CA. August 2002.
7. Barrios, A. E., “Terrain Parabolic Equation Model (TPEM) Version 1.5 User’s Manual,” Naval Command, Control and Ocean Surveillance Center, RDT&E Division, San Diego, CA, TD 2898, February 1996.
8. Space and Naval Warfare Systems Center San Diego (SSC San Diego), “Software Design Document for the Advanced Propagation Model (APM) CSCI,” in TD 3033. SSC San Diego, San Diego, CA. August 1998.

3. TEST PREPARATIONS

3.1 HARDWARE PREPARATION

Not applicable

3.2 SOFTWARE PREPARATION

A short driver program, APMMAIN.F90, is provided in Section 7. This program exercises the main software components, APMINIT CSC, APMSTEP CSC, XOINIT CSC, and XOSTEP CSC that comprise the APM CSCI. The driver program demonstrates how to access the APM CSCI and to exercise the test cases listed in the following sections. It is written to read all necessary input data for the test cases from files in a specific format. All necessary input information is presented in tabular form in Section 4.3 and the input files for each test case are listed in Section 8.

One of the main features of APM is the use of dynamic allocation in most of the arrays used for both numeric calculations and as inputs to the model. Care must be taken by the NITES CSCI application designer to properly allocate memory and initialize all variable and array inputs to APM. Ultimately, it is the responsibility of the NITES CSCI application designer to provide the necessary input in the form required by the APM CSCI.

3.3 OTHER PRETEST PREPARATION

None.

4. TEST DESCRIPTIONS

The test specification for the APM CSCI consists of 36 separate tests that exercise all subroutines and functions of the CSCI. For ease of testing, each of these 36 tests is given a name describing which portion of the APM CSCI is being exercised. All 36 tests and their descriptions are listed in Table 1.

Table 1. Test names and descriptions.

Test Name	Description
ABSORB	Gaseous absorption attenuation rate is specified.
AIRBORNE	Airborne platform for antenna height.
BLOCK	The terrain profile consists of a vertical flat-topped block or obstacle in which the terrain slope is undefined.
COSEC2	Antenna pattern is of cosecant-squared type.
EDUCT	The refractivity consists of a 14-meter evaporation duct profile.
EDUCTRF	The refractivity consists of a 14-meter evaporation duct in the presence of rough seas with wind speed of 10 m/s.
FLTA50	Raised flat land with antenna height of 50 m.
GASABS	The surface absolute humidity and surface air temperature are specified in order to compute a gaseous absorption attenuation rate.
GAUSS	Antenna pattern is of Gaussian type.
HIBW	Large vertical beamwidth is specified.
HIEL	High elevation angle is specified.
HIFREQ	High frequency.
HITRAN	High transmitter antenna height.
HORZ	Horizontal polarization antenna and standard atmosphere.
HTFIND	Antenna pattern is of generic height-finder type.
LOBW	Small vertical beamwidth is specified.
LOEL	Low elevation angle is specified.
LOFREQ	Low frequency.
LOTRAN	Low transmitter antenna height.
MPRT	Mid-path reflection over wedge.
PERW	Propagation over rounded wedge using PE model only.
PVT	Parabolic valley with short range.
RDLONGB	Range-dependent refractivity over a DTED-extracted terrain profile from Long Beach to Point Mugu, using vertical polarization and generic ground composition types.
RNGDEP	Range-dependent refractivity over smooth earth (over-water case).
SBDUCT	300-meter surface-based duct, over-water case.
SBDUCTRF	Exercises rough surface model for surface-based duct case, with wind speed of 10 m/s.
SINEX	Antenna pattern is of Sine(X)/X type.

Table 1. Test names and descriptions. (Continued)

Test Name	Description
TROPOS	Exercises troposcatter model for smooth surface (over-water case).
TROPOT	Exercises troposcatter model for terrain case.
USERDEFA	User-defined antenna pattern with explicit power and angle information.
USERHF	Antenna pattern is of specific height-finder type, with user-specified cut-back angles and power factors.
VERT	Vertical polarization antenna is specified (short-range over-water case, standard atmosphere).
VERTMIX	Vertical polarization antenna over mixed land-sea terrain path.
VERTSEA	Vertical polarization antenna is specified (long-range over-water case, ducting atmosphere).
VERTUSRD	Vertical polarization antenna and user-specified dielectric ground constants.
WEDGE	The terrain profile consists of a triangular wedge.

4.1 REQUIREMENTS ADDRESSED

Not applicable.

4.2 PREREQUISITE CONDITIONS

None.

4.3 TEST INPUTS

Although there are actual values for all input parameters listed in the input files in Section 8, some are ignored depending on the values of certain input parameters. Those input parameters that are inapplicable depending on the test case are listed as “N/A” in the tables. Note that for all test cases, the error flags *lerr6* and *lerr12* are set to “.TRUE.”. These flags allow for extra error control regarding terrain and refractivity inputs. We recommend that these error flags always be set to “.TRUE.”. However, we allowed the capability of the NITES applications designer to bypass these error controls according to the application.

The external environmental data element requirements are listed in Table 2 for each test name, with Table 3 through Table 8 providing specific height and M-unit values. The external EM system data element requirements are listed Table 9.

Table 2. External environmental data element requirements^a.

Test Name	<i>hmsl; refmsl</i> Table	n_{prof}	<i>lvlp</i>	<i>rngprof</i> ^b Table	<i>abs_{num}</i> (g/m ³)	t_{air} (°C)	γ_a (dB/km)	n_w	<i>rngwind</i> (km)	<i>wind</i> (m/s)
ABSORB	3	1	2	0.	0.	0.	.146	0	N/A	N/A
AIRBORNE	8	1	5	0.	0.	0.	0.	0	N/A	N/A
BLOCK	3	1	2	0.	0.	0.	0.	0	N/A	N/A
COSEC2	3	1	2	0.	0.	0.	0.	0	N/A	N/A
EDUCT	5	1	21	0.	0.	0.	0.	0	N/A	N/A
EDUCTRF	5	1	21	0.	0.	0.	0.	1	0.	10.
FLTA50	3	1	2	0.	0.	0.	0.	0	N/A	N/A
GASABS	3	1	2	0.	10.	25.	0.	0	N/A	N/A
GAUSS	3	1	2	0.	0.	0.	0.	0	N/A	N/A
HIBW	3	1	2	0.	0.	0.	0.	0	N/A	N/A
HIEL	3	1	2	0.	0.	0.	0.	0	N/A	N/A
HIFREQ	3	1	2	0.	0.	0.	0.	0	N/A	N/A
HITRAN	3	1	2	0.	0.	0.	0.	0	N/A	N/A
HORZ	3	1	2	0.	0.	0.	0.	0	N/A	N/A
HTFIND	3	1	2	0.	0.	0.	0.	0	N/A	N/A
LOBW	3	1	2	0.	0.	0.	0.	0	N/A	N/A
LOEL	3	1	2	0.	0.	0.	0.	0	N/A	N/A
LOFREQ	3	1	2	0.	0.	0.	0.	0	N/A	N/A
LOTRAN	3	1	2	0.	0.	0.	0.	0	N/A	N/A
MPRT	3	1	2	0.	7.5	0.	0.	0	N/A	N/A
PERW	3	1	2	0.	7.5	0.	0.	0	N/A	N/A
PVT	3	1	2	0.	7.5	0.	0.	0	N/A	N/A
RDLONGB	6	2	4	6	0.	0.	0.	0.	N/A	N/A
RNGDEP	7	2	4	7	0.	0.	0.	0.	N/A	N/A

Table 2. External environmental data element requirements^a. (Continued)

Test Name	<i>hmsl, refmsl</i> Table	<i>n_{prof}</i>	<i>lv/p</i>	<i>rngprof</i> ^b Table	<i>abs_{hum}</i> (g/m ³)	<i>t_{air}</i> (°C)	<i>γ_a</i> (dB/km)	<i>n_w</i>	<i>rngwind</i> (km)	<i>wind</i> (m/s)
SBDUCT	4	1	4	0.	0.	0.	0.	0.	N/A	N/A
SBDUCTRF	4	1	4	0.	0.	0.	0.	1.	0.	10.
SINEX	3	1	2	0.	0.	0.	0.	0.	N/A	N/A
TROPOS	3	1	2	0.	0.	0.	0.	0.	N/A	N/A
TROPOT	3	1	2	0.	0.	0.	0.	0.	N/A	N/A
USERDEFA	4	1	4	0.	0.	0.	0.	0.	N/A	N/A
USERHF	3	1	2	0.	0.	0.	0.	0.	N/A	N/A
VERT	3	1	2	0.	0.	0.	0.	0.	N/A	N/A
VERTMIX	3	1	2	0.	0.	0.	0.	0.	N/A	N/A
VERTSEA	4	1	4	0.	0.	0.	0.	0.	N/A	N/A
VERTUSRD	3	1	2	0.	0.	0.	0.	0.	N/A	N/A
WEDGE	3	1	2	0.	0.	0.	0.	0.	N/A	N/A

^aThe interpolation flag, *iextra*, is set to 0 for all test cases.

^bThe refractivity profile range is in meters except for cases RDLONGB and RNGDEP.

Table 3. Standard atmosphere with 118-M/km gradient.

i	$hmsl_{i,1}$ (meters)	$refmsl_{i,1}$ (M-unit)
1	0.	350.
2	1000.	468.

Table 4. 300-meter surface based duct atmosphere.

i	$hmsl_{i,1}$ (meters)	$refmsl_{i,1}$ (M-unit)
1	0.	339.0
2	250.	368.5
3	300.	319.0
4	1000.	401.6

Table 5. Atmosphere with 14-meter evaporation duct.

i	$hmsl_{i,1}$ (meters)	$refmsl_{i,1}$ (M-unit)
1	0.000	339.00
2	0.040	335.10
3	0.100	333.66
4	0.200	332.60
5	0.398	331.54
6	0.794	330.51
7	1.585	329.53
8	4.362	328.65
9	6.310	327.96
10	12.589	327.68
11	14.000	327.67
12	25.119	328.13
13	39.811	329.25
14	50.119	330.18
15	63.096	331.44
16	79.433	334.32
17	100.000	335.33
18	125.893	338.20
19	158.489	341.92
20	199.526	346.69
21	209.526	347.87

Table 6. Range-dependent atmosphere, standard atmosphere to surface-based duct.

i	Standard Atmosphere $rngprof_1 = 0$ km		Surface-based Duct $rngprof_2 = 100$ km	
	$hmsl_{i,1}$ (meters)	$refmsl_{i,1}$ (M-unit)	$hmsl_{i,2}$ (meters)	$refmsl_{i,2}$ (M-unit)
1	0.	350.	0.	339.0
2	0.	350.	250.	368.5
3	0.	350.	300.	319.0
4	1000.	468.	1000.	401.6

Table 7. Range-dependent atmosphere, surface-based duct to high elevated duct.

i	Surface-based Duct $rngprof_1 = 0$. km		High Elevated Duct $rngprof_2 = 250$. km	
	$hmsl_{i,1}$ meters	$refmsl_{i,1}$ M-unit	$hmsl_{i,2}$ meters	$refmsl_{i,2}$ M-unit
1	0.	330.	0.	330.
2	100.	342.5	600.	405.
3	230.	312.5	730.	375.
4	2000.	517.8	2000.	522.3

Table 8. Elevated duct.

i	$hmsl_{i,1}$ (meters)	$refmsl_{i,1}$ (M-unit)
1	0.	209.2
2	1100.	339.0
3	1500.	386.2
4	1625.	361.5
5	2100.	417.5

Table 9. External EM system data element requirements.

Test Name	f_{MHz} (MHz)	ant_{ht} (meters)	i_{pat} note a	i_{pol} note b	μ_{bw} (deg)	μ_o (deg)	n_{fac}	$hfang$ (deg)	$hffac$
ABSORB	20000.	25.	1	0	N/A	N/A	0	N/A	N/A
AIRBORNE	900.	2500.	1	0	N/A	N/A	0	N/A	N/A
BLOCK	1000.	25.	1	0	N/A	N/A	0	N/A	N/A
COSEC2	1000.	25.	4	0	1.	0.	0	N/A	N/A
EDUCT	10000.	15.	2	0	5.	0.	0	N/A	N/A
EDUCTRF	10000.	15.	2	0	5	0.	0	N/A	N/A
FLTA50	1000.	50.	1	1	N/A	N/A	0	N/A	N/A
GASABS	20000.	25.	1	0	N/A	N/A	0	N/A	N/A
GAUSS	1000.	25.	2	0	1.	0.	0	N/A	N/A
HIBW	1000.	25.	2	0	45.	0.	0	N/A	N/A
HIEL	1000.	25	2	0	1.	10.	0	N/A	N/A
HIFREQ	20000.	25.	1	0	N/A	N/A	0	N/A	N/A
HITRAN	1000.	100.	1	0	N/A	N/A	0	N/A	N/A
HORZ	1000.	25.	1	0	N/A	N/A	0	N/A	N/A
HTFIND	1000.	25.	5	0	2.	0.	0	N/A	N/A
LOBW	1000.	25.	2	0	.5	0.	0	N/A	N/A
LOEL	1000.	25.	2	0	1.	-10.	0	N/A	N/A
LOFREQ	100.	25.	1	0	N/A	N/A	0	N/A	N/A
LOTRAN	1000.	1.	1	0	N/A	N/A	0	N/A	N/A
MPRT	300.	800.	2	1	.5	-2.5	0	N/A	N/A
PERW	300	10.	1	1	N/A	N/A	0	N/A	N/A
PVT	500.	10.	1	1	N/A	N/A	0	N/A	N/A
RDLONGB	150.	100.	1	0	N/A	N/A	0	N/A	N/A
RNGDEP	3000.	25.	1	0	N/A	N/A	0	N/A	N/A
SBDUCT	3000.	25.	2	0	5.	0.	0	N/A	N/A
SBDUCTRF	3000.	25.	2	1	5.	0.	0	N/A	N/A
SINEX	1000.	25.	3	0	1.	0.	0	N/A	N/A
TROPOS	100.	25.	1	0	N/A	N/A	0	N/A	N/A
TROPOT	100.	25.	1	0	N/A	N/A	0	N/A	N/A
USERDEFA	900.	6.	7	0	N/A	2.	54	Table 10	Table 10
USERHF	1000.	25.	6	0	1.	0.	10	Table 11	Table 11
VERT	1000.	25.	1	1	N/A	N/A	0	N/A	N/A
VERTMIX	100.	10.	1	1	N/A	N/A	0	N/A	N/A
VERTSEA	100.	25.	1	1	N/A	N/A	0	N/A	N/A
VERTUSRD	100.	10.	1	1	N/A	N/A	0	N/A	N/A
WEDGE	1000.	25.	1	0	N/A	N/A	0	N/A	N/A

^aAntenna Pattern: 1=Omni-directional; 2=Gaussian; 3=Sine(X)/X; 4=Cosecant-squared; 5=Generic height-finder; 6=User-specified height finder, 7=User-defined antenna pattern.

^bPolarization: 0=Horizontal; 1=Vertical

Table 10. Height-finder angles and factors for case USERDEFA.

i	$hfang_i$ (deg)	$hffac_i$
1	-17	.017
2	-16	.044
3	-15	.080
4	-14	.126
5	-13	.182
6	-12	.245
7	-11	.316
8	-10	.389
9	-9	.479
10	-8	.556
11	-7	.631
12	-6	.716
13	-5	.785
14	-4	.861
15	-3	.912
16	-2	.966
17	-1	.998
18	0	1.00
19	1	1.00
20	2	.966
21	3	.902
22	4	.822
23	5	.742
24	6	.646
25	7	.569
26	8	.501
27	9	.452
28	10	.422
29	11	.402
30	12	.389
31	13	.375
32	14	.359
33	15	.339
34	16	.305
35	17	.276
36	18	.245
37	19	.221
38	20	.210
39	21	.199
40	22	.190
41	23	.180

Table 10. Antenna pattern angles and factors for case USERDEFA. (Continued)

i	$hfang_i$ (deg)	$hffac_i$
42	24	.164
43	25	.148
44	26	.130
45	27	.110
46	28	.095
47	29	.077
48	30	.070
49	31	.065
50	32	.058
51	33	.050
52	34	.039
53	35	.031
54	36	.025

Table 11. Height-finder angles and factors for case USERHF.

i	$hfang_i$ (deg)	$hffac_i$
1	1.0	0.9
2	1.5	0.8
3	2.0	0.7
4	2.5	0.6
5	3.0	0.5
6	3.5	0.4
7	4.0	0.3
8	4.5	0.2
9	5.0	0.1
10	5.5	0.0

The external implementation data element requirements that must be specified for each test are listed in Table 12.

Table 12. External implementation data element requirements ^a.

Test Name	h_{min} (meters)	h_{max} (meters)	n_{rout}	n_{zout}	PE_{flag} ^b	r_{max} (km)	r_{mult}	th_{max} (deg)	T_{ropo} ^b
ABSORB	0.	200.	1	20	F	50.	N/A	N/A	F
AIRBORNE	0.	5000.	1	20	F	250.	N/A	N/A	F
BLOCK	0.	400.	1	20	F	60.	N/A	N/A	F
COSEC2	0.	2000.	1	20	F	50.	N/A	N/A	F
EDUCT	0.	200.	1	20	F	50.	N/A	N/A	F
EDUCTRF	0.	200.	1	20	F	100.	N/A	N/A	F
FLTA50	0.	100.	1	20	F	50.	N/A	N/A	F
GASABS	0.	200.	1	20	F	50.	N/A	N/A	F
GAUSS	0.	2000.	1	20	F	50.	N/A	N/A	F
HIBW	0.	2000.	1	20	F	50.	N/A	N/A	F
HIEL	0.	20,000.	1	20	F	50.	N/A	N/A	F
HIFREQ	0.	200.	1	20	F	50.	N/A	N/A	F
HITRAN	0.	1000.	1	20	F	50.	N/A	N/A	F
HORZ	0.	2000.	1	20	F	50.	N/A	N/A	F
HTFIND	0.	2000.	1	20	F	50.	N/A	N/A	F
LOBW	0.	2000.	1	20	F	50.	N/A	N/A	F
LOEL	0.	20,000.	1	20	F	50.	N/A	N/A	F
LOFREQ	0.	5000.	1	20	F	50.	N/A	N/A	F
LOTRAN	0.	10,000.	1	20	F	50.	N/A	N/A	F
MPRT	0.	1100.	30	1	F	60.	N/A	N/A	F
PERW	0.	1000.	20	1	T	50.	1.	10.	F
PVT	0.	2000.	1	20	F	10.	N/A	N/A	F
RDLONGB	0.	1000.	1	20	F	100.	N/A	N/A	F
RNGDEP	0.	2000.	1	20	F	250.	N/A	N/A	F
SBDUCT	0.	5000.	1	20	F	200.	N/A	N/A	F
SBDUCTRF	0.	1000.	1	20	F	200.	N/A	N/A	F
SINEX	0.	2000.	1	20	F	50.	N/A	N/A	F
TROPOS	0.	2000.	1	20	F	200.	N/A	N/A	T
TROPOT	0.	2000.	1	20	F	200.	N/A	N/A	T
USERDEFA	0.	3000.	1	20	F	300.	N/A	N/A	F
USERHF	0.	2000.	1	20	F	50.	N/A	N/A	F
VERT	0.	2000.	1	20	F	50.	N/A	N/A	F
VERTMIX	0.	1000.	1	20	F	50.	N/A	N/A	F
VERTSEA	0.	1000.	1	20	F	300.	N/A	N/A	F
VERTUSRD	0.	1000.	1	20	F	50.	N/A	N/A	F
WEDGE	0.	1000.	1	20	F	100.	N/A	N/A	F

^aLogical flags *lerr6* and *lerr12* are 'true.' for all cases.^b'T' = 'true.'; 'F' = 'false.'

The external terrain data element requirements are listed in Table 13. Terrain profiles used for specific test cases are listed in Table 14 through Table 23.

Table 13. External terrain data element requirements.

Test Name	<i>terx, tery</i> Table	<i>i_{tp}</i>	<i>i_{gr}</i>	<i>igrnd</i>	<i>rgrnd</i> (km)	<i>dielec</i> (ϵ_r, σ) ^a
ABSORB	N/A	N/A	N/A	N/A	N/A	N/A
BLOCK	Table 14	6	1	7	0.	(7.5, .01)
COSEC2	N/A	N/A	N/A	N/A	N/A	N/A
EDUCT	N/A	N/A	N/A	N/A	N/A	N/A
EDUCTRF	N/A	N/A	N/A	N/A	N/A	N/A
FLTA50	Table 15	2	1	7	0.	(7.0, .01)
GASABS	N/A	N/A	N/A	N/A	N/A	N/A
GAUSS	N/A	N/A	N/A	N/A	N/A	N/A
HIBW	N/A	N/A	N/A	N/A	N/A	N/A
HIEL	N/A	N/A	N/A	N/A	N/A	N/A
HIFREQ	N/A	N/A	N/A	N/A	N/A	N/A
HITRAN	N/A	N/A	N/A	N/A	N/A	N/A
HORZ	N/A	N/A	N/A	N/A	N/A	N/A
HTFIND	N/A	N/A	N/A	N/A	N/A	N/A
LOBW	N/A	N/A	N/A	N/A	N/A	N/A
LOEL	N/A	N/A	N/A	N/A	N/A	N/A
LOFREQ	N/A	N/A	N/A	N/A	N/A	N/A
LOTRAN	N/A	N/A	N/A	N/A	N/A	N/A
MPRT	Table 16	5	1	7	0.	(7.5, 0.1)
PERW	Table 17	11	1	7	0.	(7.5, 0.1)
PVT	Table 18	17	1	7	0.	(7.5, 0.1)
RDLONGB	Table 19	167	6	Table 20	Table 20	N/A
RNGDEP	N/A	N/A	N/A	N/A	N/A	N/A
SBDUCT	N/A	N/A	N/A	N/A	N/A	N/A
SBDUCTRF	N/A	N/A	N/A	N/A	N/A	N/A
SINEX	N/A	N/A	N/A	N/A	N/A	N/A
TROPOS	N/A	N/A	N/A	N/A	N/A	N/A
TROPOT	Table 19	167	6	Table 20	Table 20	N/A
USERHF	N/A	N/A	N/A	N/A	N/A	N/A
VERT	N/A	N/A	1	0	0.	N/A
VERTMIX	Table 21	2	2	Table 21	Table 21	N/A
VERTSEA	N/A	N/A	N/A	N/A	N/A	N/A
VERTUSRD	Table 22	2	1	7	0.	(3., 6e-4)
WEDGE	Table 23	5	1	0	0.	N/A

^a ϵ_r = relative permittivity; σ = conductivity (S/m)

Table 14. Terrain profile for test case BLOCK.

i	$terx_i$ (km)	$tery_i$ (meters)
1	0.	1.
2	10.	1.
3	10.	200.
4	40.	200.
5	40.	1
6	60.	1

Table 15. Terrain profile for test case FLTA50.

i	$terx_i$ (km)	$tery_i$ (meters)
1	0.	10.
2	50.	10.

Table 16. Terrain profile for test case MPRT.

i	$terx_i$ (km)	$tery_i$ (meters)
1	0.	0.
2	10.	0.
3	30.	600.
4	50.	0.
5	60.	0.

Table 17. Terrain profile for test case PERW.

i	$terx_i$ (km)	$tery_i$ (meters)
1	0.	0.
2	18.75	0.
3	20.312	210.
4	21.875	320.
5	23.4375	375.
6	25.00	390.
7	26.565	375.
8	28.125	320.
9	31.250	90.
10	32.8125	0.
11	50.00	0.

Table 18. Terrain profile for test case PVT.

i	$terx_i$ (km)	$tery_i$ (meters)
1	0.00	625.
2	3.17	476.
3	6.34	347.
4	9.51	239.
5	12.69	151.
6	15.87	83.
7	19.04	35.
8	22.22	7.
9	25.00	0.
10	27.78	7.
11	30.96	35.
12	34.13	83.
13	37.13	151.
14	40.49	239.
15	43.66	347.
16	46.83	476.
17	50.	625.

Table 19. Terrain profile for test case RDLONGB and TROPOT.

i	$terx_i$ (km)	$tery_i$ (meters)	i	$terx_i$ (km)	$tery_i$ (meters)	i	$terx_i$ (km)	$tery_i$ (meters)
1	0.0	8.0	57	20.10	22.0	113	79.20	184.0
2	.30	8.0	58	20.40	23.0	114	79.50	226.0
3	.60	9.0	59	20.70	24.0	115	79.80	152.0
4	.90	9.0	60	21.00	24.0	116	80.10	201.0
5	1.20	10.0	61	21.30	25.0	117	80.40	244.0
6	1.50	11.0	62	21.60	26.0	118	80.70	152.0
7	1.80	12.0	63	21.90	27.0	119	81.00	143.0
8	2.10	13.0	64	22.20	27.0	120	81.30	91.0
9	2.40	14.0	65	22.50	28.0	121	81.60	107.0
10	2.70	15.0	66	22.80	29.0	122	81.90	152.0
11	3.00	17.0	67	23.40	29.0	123	82.20	152.0
12	3.30	19.0	68	23.70	30.0	124	82.50	170.0
13	3.60	21.0	69	24.60	30.0	125	82.80	152.0
14	3.90	23.0	70	24.90	32.0	126	83.10	66.0
15	4.20	25.0	71	25.20	34.0	127	83.40	70.0
16	4.50	27.0	72	25.50	38.0	128	83.70	121.0
17	4.80	28.0	73	26.10	38.0	129	84.00	152.0

Table 19. Terrain profile for test case RDLONGB and TROPOT. (Continued)

i	$terx_i$ (km)	$tery_i$ (meters)	i	$terx_i$ (km)	$tery_i$ (meters)	i	$terx_i$ (km)	$tery_i$ (meters)
18	5.10	30.0	74	26.40	36.0	130	84.30	170.0
19	5.40	31.0	75	26.70	34.0	131	84.60	141.0
20	5.70	31.0	76	27.00	32.0	132	84.90	139.0
21	6.00	29.0	77	27.30	27.0	133	85.20	147.0
22	6.30	23.0	78	27.60	15.0	134	85.50	177.0
23	6.60	14.0	79	27.90	6.0	135	85.80	152.0
24	6.90	9.0	80	28.20	1.0	136	86.10	61.0
25	7.20	7.0	81	28.50	0.0	137	86.70	61.0
26	7.50	7.0	82	64.50	0.0	138	87.00	70.0
27	7.80	9.0	83	64.80	8.0	139	87.30	44.0
28	8.10	11.0	84	65.10	30.0	140	87.60	11.0
29	8.40	14.0	85	65.40	39.0	141	87.90	1.0
30	8.70	13.0	86	65.70	61.0	142	89.40	1.0
31	9.30	13.0	87	66.60	61.0	143	89.70	61.0
32	9.60	12.0	88	66.90	24.0	144	90.00	84.0
33	9.90	11.0	89	67.20	14.0	145	90.30	152.0
34	10.20	8.0	90	67.50	26.0	146	90.60	152.0
35	10.80	8.0	91	67.80	16.0	147	90.90	101.0
36	11.10	7.0	92	68.10	1.0	148	91.20	40.0
37	12.60	7.0	93	68.40	1.0	149	91.50	15.0
38	12.90	6.0	94	68.70	0.0	150	91.80	20.0
39	14.40	6.0	95	73.80	0.0	151	92.10	2.0
40	14.70	7.0	96	74.10	1.0	152	92.40	10.0
41	15.00	8.0	97	74.40	1.0	153	92.70	4.0
42	15.30	8.0	98	74.70	10.0	154	93.00	1.0
43	15.60	9.0	99	75.00	8.0	155	93.30	1.0
44	15.90	10.0	100	75.30	39.0	156	93.60	0.0
45	16.20	11.0	101	75.60	45.0	157	93.90	1.0
46	16.50	11.0	102	75.90	53.0	158	96.30	1.0
47	16.80	12.0	103	76.20	61.0	159	96.60	0.0
48	17.40	12.0	104	76.50	61.0	160	96.90	1.0
49	17.70	13.0	105	76.80	82.0	161	97.50	1.0
50	18.00	13.0	106	77.10	61.0	162	97.80	2.0
51	18.30	14.0	107	77.40	78.0	163	98.10	3.0
52	18.60	15.0	108	77.70	61.0	164	99.30	3.0
53	18.90	16.0	109	78.00	129.0	165	99.60	2.0
54	19.20	18.0	110	78.30	30.0	166	99.90	2.0
55	19.50	20.0	111	78.60	46.0	167	100.20	1.0
56	19.80	21.0	112	78.90	159.0			

Table 20. Table of ground constants for terrain profile of Table 19.

i_{gr}	$igrnd_i^a$	$rgrnd_i$ (km)
1	2	0.
2	0	28.5
3	3	64.8
4	0	68.7
5	4	74.1
6	0	100.2

^aGround composition type: 0=sea water; 1=fresh water; 2=wet ground; 3=medium dry ground; 4=very dry ground; 5=ice at -1°C; 6=ice at -10°C; 7=user-defined permittivity and conductivity.

Table 21. Terrain profile for test case VERTMIX.

i	$terx_i$ (km)	$tery_i$ meters	i_{gr}	$igrnd_i^a$	$rgrnd_i$ (km)
1	0.	0.	1	4	0.
2	50.	0.	2	0	25.0

^aGround composition type: 0=sea water; 1=fresh water; 2=wet ground; 3=medium dry ground; 4=very dry ground; 5=ice at -1°C; 6=ice at -10°C; 7=user-defined permittivity and conductivity.

Table 22. Terrain profile for test case VERTUSRD.

i	$terx_i$ (km)	$tery_i$ (meters)
1	0.	0.
2	50.	0.

Table 23. Terrain profile for test case WEDGE.

i	$terx_i$ (km)	$tery_i$ (meters)
1	0.	0.
2	45.0	0.
3	50.0	200.
4	55.0	0.
5	100.0	0.

4.4 EXPECTED TEST RESULTS

The expected test result propagation loss versus height values for each of the 36 test cases are listed in tabular form within Table 24 through Table 59.

Table 24. Expected output for ABSORB for r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
10.00	212.70	-60.20
20.00	199.30	-46.80
30.00	188.90	-36.50
40.00	180.10	-27.60
50.00	172.20	-19.80
60.00	165.50	-13.00
70.00	160.10	-7.60
80.00	156.70	-4.30
90.00	156.50	-4.00
100.00	163.20	-10.70
110.00	159.30	-6.90
120.00	156.00	-3.60
130.00	167.80	-15.30
140.00	155.70	-3.30
150.00	163.00	-10.50
160.00	156.10	-3.60
170.00	161.90	-9.40
180.00	155.70	-3.30
190.00	164.50	-12.00
200.00	154.90	-2.50

Table 25. Expected output for AIRBORNE for r_{max} receiver range of 250 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
250.00	136.90	2.60
500.00	136.00	3.50
750.00	138.00	1.50
1000.00	139.90	-0.40
1250.00	138.10	1.40
1500.00	132.90	6.50
1750.00	142.70	-3.30
2000.00	142.40	-2.90
2250.00	143.10	-3.60
2500.00	142.20	-2.70
2750.00	140.70	-1.20
3000.00	136.40	3.10
3250.00	143.50	-4.00
3500.00	142.90	-3.40
3750.00	137.90	1.60
4000.00	143.00	-3.50
4250.00	138.70	0.80
4500.00	142.80	-3.30
4750.00	143.60	-4.20
5000.00	137.40	2.10

Table 26. Expected output for BLOCK
for r_{max} receiver range of 60 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
20.00	216.10	-88.10
40.00	209.40	-81.40
60.00	207.80	-79.80
80.00	209.00	-81.00
100.00	207.20	-79.20
120.00	203.00	-75.00
140.00	200.70	-72.70
160.00	198.60	-70.60
180.00	195.10	-67.00
200.00	191.60	-63.60
220.00	188.20	-60.20
240.00	184.40	-56.40
260.00	180.60	-52.60
280.00	176.90	-48.90
300.00	173.30	-45.30
320.00	170.00	-42.00
340.00	167.10	-39.10
360.00	164.70	-36.70
380.00	162.60	-34.60
400.00	160.70	-32.70

Table 27. Expected Output for COSEC2
for r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
100.0	134.40	-8.00
200.0	124.10	2.30
300.0	122.30	4.10
400.0	129.70	-3.30
500.0	126.50	-0.10
600.0	123.50	2.90
700.0	128.00	-1.60
800.0	126.90	-0.40
900.0	125.70	0.70
1000.0	126.60	-0.20
1100.0	127.10	-0.60
1200.0	127.50	-1.10
1300.0	128.90	-2.40
1400.0	129.50	-3.10
1500.0	129.70	-3.30
1600.0	131.00	-4.50
1700.0	131.50	-5.00
1800.0	131.40	-4.90
1900.0	132.70	-6.20
2000.0	133.00	-6.60

Table 28. Expected Output for EDUCT
for r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
10.0	142.80	3.60
20.0	147.50	-1.10
30.0	150.10	-3.60
40.0	152.50	-6.10
50.0	156.00	-9.60
60.0	158.60	-12.10
70.0	154.10	-7.70
80.0	149.50	-3.10
90.0	146.30	0.10
100.0	144.30	2.20
110.0	143.10	3.40
120.0	142.70	3.70
130.0	143.20	3.20
140.0	145.10	1.30
150.0	149.50	-3.10
160.0	161.60	-15.10
170.0	151.90	-5.50
180.0	145.20	1.20
190.0	142.40	4.10
200.0	141.50	4.90

Table 29. Expected output for
EDUCTRF for r_{max} receiver range of 100
km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
10.0	154.70	-2.20
20.0	157.90	-5.50
30.0	159.70	-7.30
40.0	160.30	-7.90
50.0	160.70	-8.30
60.0	161.00	-8.50
70.0	161.10	-8.60
80.0	161.20	-8.80
90.0	161.30	-8.90
100.0	161.30	-8.90
110.0	161.40	-9.00
120.0	161.50	-9.00
130.0	161.50	-9.10
140.0	161.60	-9.20
150.0	161.70	-9.30
160.0	161.80	-9.30
170.0	161.90	-9.40
180.0	162.00	-9.60
190.0	162.10	-9.70
200.0	162.30	-9.80

Table 30. Expected output for FLTA50
for r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
5.00	-999.0	-999.0
10.00	190.3	-63.9
15.00	159.0	-32.5
20.00	152.7	-26.3
25.00	148.9	-22.5
30.00	146.1	-19.7
35.00	143.8	-17.3
40.00	141.8	-15.3
45.00	140.0	-13.6
50.00	138.4	-12.0
55.00	137.0	-10.5
60.00	135.6	-9.2
65.00	134.4	-8.0
70.00	133.3	-6.8
75.00	132.2	-5.8
80.00	131.2	-4.7
85.00	130.2	-3.8
90.00	129.4	-2.9
95.00	128.5	-2.1
100.00	127.8	-1.3

Table 31. Expected output for GASABS
for r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
10.0	212.7	-60.20
20.0	199.2	-46.80
30.0	188.9	-36.50
40.0	180.1	-27.60
50.0	172.2	-19.80
60.0	165.5	-13.00
70.0	160.1	-7.60
80.0	156.7	-4.20
90.0	156.5	-4.00
100.0	163.1	-10.70
110.0	159.3	-6.90
120.0	156.0	-3.60
130.0	167.8	-15.30
140.0	155.7	-3.30
150.0	163.0	-10.50
160.0	156.1	-3.60
170.0	161.9	-9.40
180.0	155.7	-3.30
190.0	164.4	-12.00
200.0	154.9	-2.50

Table 32. Expected output for GAUSS
for r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
100.0	133.7	-7.2
200.0	123.5	2.9
300.0	121.7	4.8
400.0	130.7	-4.3
500.0	127.1	-0.6
600.0	124.0	2.4
700.0	133.0	-6.6
800.0	132.2	-5.7
900.0	129.6	-3.2
1000.0	139.0	-12.6
1100.0	140.0	-13.5
1200.0	138.1	-11.7
1300.0	148.3	-21.9
1400.0	150.4	-23.9
1500.0	149.5	-23.1
1600.0	160.5	-34.1
1700.0	163.6	-37.1
1800.0	163.7	-37.3
1900.0	175.5	-49.1
2000.0	179.6	-53.1

Table 33. Expected output for HIBW for
 r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
100.0	133.6	-7.2
200.0	123.3	3.1
300.0	121.1	5.3
400.0	129.7	-3.3
500.0	124.9	1.5
600.0	120.6	5.8
700.0	128.1	-1.7
800.0	125.3	1.1
900.0	120.5	5.9
1000.0	127.6	-1.1
1100.0	125.6	0.8
1200.0	120.5	6.0
1300.0	127.5	-1.1
1400.0	125.6	0.8
1500.0	120.5	6.0
1600.0	127.5	-1.0
1700.0	125.6	0.8
1800.0	120.5	5.9
1900.0	127.4	-1.0
2000.0	125.7	0.8

Table 34. Expected output for HIEL for r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
1000.0	376.4	-250.0
2000.0	376.4	-250.0
3000.0	376.4	-250.0
4000.0	376.4	-250.0
5000.0	364.2	-237.7
6000.0	258.9	-132.4
7000.0	184.7	-58.1
8000.0	140.8	-14.2
9000.0	126.6	0.0
10000.0	141.2	-14.6
11000.0	183.7	-57.1
12000.0	253.1	-126.4
13000.0	348.2	-221.5
14000.0	376.7	-250.0
15000.0	376.8	-250.0
16000.0	376.8	-250.0
17000.0	376.9	-250.0
18000.0	376.9	-250.0
19000.0	377.0	-250.0
20000.0	377.1	-250.0

Table 35. Expected output for HIFREQ for r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
10.0	205.4	-52.9
20.0	192.0	-39.5
30.0	181.6	-29.2
40.0	172.8	-20.3
50.0	164.9	-12.5
60.0	158.2	-5.7
70.0	152.8	-0.3
80.0	149.4	3.0
90.0	149.2	3.3
100.0	155.9	-3.4
110.0	152.0	0.4
120.0	148.7	3.7
130.0	160.5	-8.0
140.0	148.4	4.0
150.0	155.7	-3.2
160.0	148.8	3.7
170.0	154.6	-2.1
180.0	148.4	4.0
190.0	157.2	-4.7
200.0	147.6	4.8

Table 36. Expected output for HITRAN
for r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
50.0	126.3	0.1
100.0	121.8	4.7
150.0	138.1	-11.7
200.0	121.4	5.0
250.0	134.6	-8.2
300.0	122.0	4.4
350.0	124.4	2.0
400.0	127.7	-1.3
450.0	120.9	5.5
500.0	131.4	-5.0
550.0	123.2	3.3
600.0	121.5	5.0
650.0	147.9	-21.5
700.0	121.7	4.7
750.0	122.3	4.1
800.0	137.7	-11.3
850.0	121.1	5.3
900.0	123.2	3.2
950.0	132.5	-6.1
1000.0	120.8	5.6

Table 37. Expected output for HORZ for
 r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
100.0	133.6	-7.2
200.0	123.3	3.1
300.0	121.1	5.3
400.0	129.7	-3.3
500.0	124.9	1.5
600.0	120.6	5.8
700.0	128.1	-1.7
800.0	125.3	1.1
900.0	120.5	5.9
1000.0	127.6	-1.1
1100.0	125.6	0.9
1200.0	120.5	6.0
1300.0	127.5	-1.0
1400.0	125.6	0.8
1500.0	120.5	6.0
1600.0	127.4	-1.0
1700.0	125.6	0.8
1800.0	120.5	6.0
1900.0	127.4	-1.0
2000.0	125.6	0.8

Table 38. Expected output for HTFIND
for r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
100.0	133.6	-7.2
200.0	123.4	3.0
300.0	121.5	4.9
400.0	130.0	-3.6
500.0	125.8	0.7
600.0	122.1	4.3
700.0	128.8	-2.3
800.0	126.9	-0.5
900.0	124.4	2.0
1000.0	127.1	-0.7
1100.0	126.5	-0.1
1200.0	126.2	0.3
1300.0	126.6	-0.2
1400.0	126.5	-0.1
1500.0	126.2	0.3
1600.0	126.6	-0.2
1700.0	126.5	-0.1
1800.0	126.2	0.3
1900.0	126.6	-0.2
2000.0	126.5	-0.1

Table 39 Expected output for LOBW for
 r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
100.0	133.8	-7.4
200.0	124.0	2.4
300.0	123.2	3.2
400.0	132.9	-6.4
500.0	133.0	-6.6
600.0	134.1	-7.7
700.0	146.4	-19.9
800.0	151.8	-25.3
900.0	156.6	-30.2
1000.0	171.5	-45.1
1100.0	181.5	-55.1
1200.0	190.5	-64.0
1300.0	208.4	-81.9
1400.0	222.4	-96.0
1500.0	235.6	-109.2
1600.0	256.6	-130.1
1700.0	274.6	-148.2
1800.0	292.0	-165.6
1900.0	316.2	-189.7
2000.0	338.1	-211.7

Table 40. Expected output for LOEL for r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
1000.0	376.4	-250.0
2000.0	376.4	-250.0
3000.0	376.4	-250.0
4000.0	376.4	-250.0
5000.0	358.3	-231.8
6000.0	254.6	-128.2
7000.0	181.9	-55.4
8000.0	139.6	-13.1
9000.0	126.9	-0.3
10000.0	142.9	-16.3
11000.0	186.8	-60.2
12000.0	257.6	-130.9
13000.0	354.0	-227.3
14000.0	376.7	-250.0
15000.0	376.8	-250.0
16000.0	376.8	-250.0
17000.0	376.9	-250.0
18000.0	376.9	-250.0
19000.0	377.0	-250.0
20000.0	377.1	-250.0

Table 41. Expected output for LOFREQ for r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
250.0	116.7	-10.3
500.0	109.0	-2.6
750.0	105.0	1.5
1000.0	102.6	3.8
1250.0	101.2	5.2
1500.0	100.5	5.9
1750.0	100.5	5.9
2000.0	101.0	5.4
2250.0	102.3	4.1
2500.0	104.5	1.9
2750.0	108.4	-1.9
3000.0	116.9	-10.5
3250.0	119.4	-13.0
3500.0	109.2	-2.7
3750.0	104.9	1.5
4000.0	102.6	3.9
4250.0	101.2	5.2
4500.0	100.6	5.9
4750.0	100.5	6.0
5000.0	101.0	5.5

Table 42. Expected output for LOTRAN
for r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
500.0	133.5	-7.1
1000.0	126.3	0.2
1500.0	122.9	3.5
2000.0	121.2	5.3
2500.0	120.5	6.0
3000.0	120.7	5.8
3500.0	121.8	4.6
4000.0	124.1	2.3
4500.0	128.5	-2.0
5000.0	140.4	-14.0
5500.0	134.2	-7.7
6000.0	126.7	-0.2
6500.0	123.2	3.3
7000.0	121.4	5.1
7500.0	120.7	5.8
8000.0	120.8	5.8
8500.0	121.7	4.9
9000.0	123.6	3.0
9500.0	127.1	-0.6
10000.0	134.8	-8.2

Table 43. Expected output for MPRT for
 h_{max} receiver height of 1100 m.

Range (km)	Prop. Loss (dB)	Prop. Factor (dB)
2.0	295.0	-207.0
4.0	298.0	-204.0
6.0	299.8	-202.2
8.0	301.0	-201.0
10.0	302.0	-200.0
12.0	302.8	-199.2
14.0	264.9	-160.0
16.0	261.5	-155.4
18.0	265.4	-158.3
20.0	206.0	-98.0
22.0	148.8	-40.0
24.0	123.3	-13.7
26.0	121.8	-11.5
28.0	134.7	-23.8
30.0	156.7	-45.1
32.0	180.9	-68.8
34.0	207.1	-94.5
36.0	233.7	-120.6
38.0	244.2	-130.6
40.0	245.5	-131.5
42.0	246.9	-132.5
44.0	249.1	-134.3
46.0	252.3	-137.1
48.0	256.9	-141.3
50.0	261.8	-145.8
52.0	267.1	-150.8
54.0	272.0	-155.3
56.0	276.0	-159.0
58.0	279.6	-162.3
60.0	282.2	-164.7

Table 44. Expected output for PERW for h_{max} receiver height of 1000 m.

Range (km)	Prop. Loss (dB)	Prop. Factor (dB)
2.5	179.8	-89.9
5.0	129.6	-33.6
7.5	95.9	3.6
10.0	110.7	-8.7
12.5	99.7	4.2
15.0	102.4	3.1
17.5	110.2	-3.3
20.0	119.7	-11.6
22.5	111.8	-2.8
25.0	108.7	1.2
27.5	108.1	2.7
30.0	106.9	4.6
32.5	107.6	4.6
35.0	109.1	3.8
37.5	109.3	4.2
40.0	110.3	3.8
42.5	109.8	4.7
45.0	109.5	5.6
47.5	109.1	6.4
50.0	111.9	4.1

Table 45. Expected output for PVT for r_{max} receiver range of 10 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
100.0	-999.0	-999.0
200.0	-999.0	-999.0
300.0	101.3	5.1
400.0	113.5	-7.1
500.0	103.9	2.6
600.0	105.2	1.3
700.0	109.5	-3.1
800.0	104.2	2.3
900.0	107.0	-0.5
1000.0	107.7	-1.3
1100.0	106.2	0.3
1200.0	105.8	0.7
1300.0	102.9	3.6
1400.0	109.4	-3.0
1500.0	103.9	2.5
1600.0	107.7	-1.3
1700.0	103.4	3.0
1800.0	105.0	1.4
1900.0	107.7	-1.2
2000.0	103.7	2.8

Table 46. Expected output for RDLONGB for r_{max} receiver range of 100 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
50.0	138.7	-22.8
100.0	133.1	-17.1
150.0	129.3	-13.3
200.0	126.4	-10.5
250.0	126.8	-10.8
300.0	129.5	-13.5
350.0	125.0	-9.1
400.0	121.7	-5.7
450.0	119.1	-3.2
500.0	118.5	-2.5
550.0	118.5	-2.5
600.0	117.1	-1.1
650.0	114.6	1.4
700.0	112.6	3.4
750.0	113.6	2.4
800.0	112.6	3.4
850.0	111.0	5.0
900.0	110.9	5.1
950.0	110.9	5.1
1000.0	110.1	5.8

Table 47. Expected output for RNGDEP for r_{max} receiver range of 250 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
100.0	199.8	-49.9
200.0	195.8	-45.9
300.0	202.8	-52.8
400.0	178.6	-28.7
500.0	141.9	8.0
600.0	135.4	14.5
700.0	150.9	-0.9
800.0	164.2	-14.2
900.0	166.8	-16.9
1000.0	182.9	-33.0
1100.0	196.7	-46.8
1200.0	197.4	-47.4
1300.0	200.7	-50.8
1400.0	195.1	-45.2
1500.0	192.9	-43.0
1600.0	191.5	-41.5
1700.0	192.4	-42.4
1800.0	195.5	-45.5
1900.0	194.5	-44.6
2000.0	193.3	-43.3

Table 48. Expected output for SBDUCT
for r_{max} receiver range of 200 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
250.00	139.9	8.1
500.00	166.0	-18.0
750.00	157.0	-9.0
1000.0	161.3	-13.3
1250.0	174.6	-26.6
1500.0	169.5	-21.4
1750.0	158.6	-10.6
2000.0	150.8	-2.8
2250.0	146.9	1.1
2500.0	147.7	0.3
2750.0	165.7	-17.7
3000.0	145.1	3.0
3250.0	148.0	0.0
3500.0	147.0	1.0
3750.0	145.3	2.7
4000.0	149.7	-1.7
4250.0	144.4	3.7
4500.0	149.9	-1.9
4750.0	144.5	3.5
5000.0	148.0	0.0

Table 49. Expected output for
SBDUCTRF for r_{max} receiver range of
200 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
50.0	141.8	6.2
100.0	136.6	11.5
150.0	141.5	6.5
200.0	141.3	6.7
250.0	141.0	7.0
300.0	152.3	-4.3
350.0	171.6	-23.6
400.0	182.3	-34.3
450.0	172.7	-24.7
500.0	168.7	-20.7
550.0	165.4	-17.4
600.0	162.9	-14.9
650.0	161.0	-12.9
700.0	159.9	-11.9
750.0	159.1	-11.1
800.0	159.1	-11.0
850.0	160.0	-12.0
900.0	160.1	-12.1
950.0	161.1	-13.1
1000.0	162.7	-14.7

Table 50. Expected output for SINEX for r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
100.0	133.7	-7.2
200.0	123.5	3.0
300.0	121.6	4.8
400.0	130.7	-4.3
500.0	127.0	-0.6
600.0	124.1	2.4
700.0	133.3	-6.8
800.0	133.0	-6.6
900.0	131.9	-5.4
1000.0	143.2	-16.8
1100.0	151.3	-24.8
1200.0	150.9	-24.5
1300.0	157.9	-31.5
1400.0	156.1	-29.6
1500.0	150.9	-24.5
1600.0	157.9	-31.5
1700.0	156.1	-29.7
1800.0	150.9	-24.5
1900.0	157.9	-31.5
2000.0	156.1	-29.7

Table 51. Expected output for TROPOS for r_{max} receiver range of 200 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
100.0	165.2	-46.7
200.0	164.6	-46.1
300.0	164.5	-46.0
400.0	164.4	-46.0
500.0	164.4	-46.0
600.0	164.2	-45.7
700.0	163.4	-45.0
800.0	162.1	-43.6
900.0	161.2	-42.8
1000.0	158.5	-40.0
1100.0	155.9	-37.4
1200.0	153.5	-35.0
1300.0	151.2	-32.7
1400.0	149.0	-30.5
1500.0	146.9	-28.4
1600.0	144.9	-26.5
1700.0	143.1	-24.6
1800.0	141.3	-22.8
1900.0	139.6	-21.1
2000.0	138.0	-19.6

Table 52. Expected output for TROPOT
for r_{max} receiver range of 200 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
100.0	164.7	-46.2
200.0	163.7	-45.2
300.0	162.8	-44.3
400.0	161.2	-42.7
500.0	159.2	-40.7
600.0	157.1	-38.6
700.0	155.2	-36.7
800.0	153.6	-35.1
900.0	152.4	-34.0
1000.0	151.7	-33.2
1100.0	151.1	-32.7
1200.0	150.6	-32.1
1300.0	149.6	-31.1
1400.0	147.9	-29.4
1500.0	145.9	-27.4
1600.0	143.8	-25.4
1700.0	142.1	-23.6
1800.0	140.5	-22.0
1900.0	139.0	-20.5
2000.0	137.3	-18.9

Table 53. Expected output for
USERDEFA for r_{max} receiver range of
300 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
150.0	137.7	3.3
300.0	138.8	2.3
450.0	159.0	-17.9
600.0	153.2	-12.2
750.0	151.3	-10.2
900.0	152.6	-11.5
1050.0	154.8	-13.8
1200.0	158.9	-17.9
1350.0	168.6	-27.5
1500.0	165.0	-24.0
1650.0	156.5	-15.4
1800.0	153.6	-12.5
1950.0	153.0	-11.9
2100.0	152.2	-11.1
2250.0	151.0	-10.0
2400.0	151.0	-10.0
2550.0	151.8	-10.7
2700.0	152.5	-11.4
2850.0	153.7	-12.6
3000.0	155.3	-14.2

Table 54. Expected output for USERHF for r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
100.00	133.6	-7.2
200.00	123.8	2.6
300.00	122.6	3.8
400.00	128.9	-2.5
500.00	126.7	-0.3
600.00	126.0	0.4
700.00	126.6	-0.2
800.00	126.5	-0.1
900.00	126.2	0.3
1000.0	126.6	-0.2
1100.0	127.4	-1.0
1200.0	127.1	-0.7
1300.0	127.5	-1.1
1400.0	127.4	-1.0
1500.0	128.1	-1.7
1600.0	128.5	-2.1
1700.0	128.5	-2.0
1800.0	128.1	-1.7
1900.0	128.5	-2.1
2000.0	129.6	-3.2

Table 55. Expected output for VERT for r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
100.0	133.8	-7.3
200.0	123.5	2.9
300.0	121.4	5.1
400.0	129.3	-2.9
500.0	125.9	0.5
600.0	121.3	5.1
700.0	127.8	-1.4
800.0	127.0	-0.6
900.0	121.7	4.8
1000.0	127.2	-0.8
1100.0	127.9	-1.5
1200.0	122.1	4.4
1300.0	126.9	-0.5
1400.0	128.5	-2.0
1500.0	122.5	4.0
1600.0	126.7	-0.3
1700.0	128.8	-2.4
1800.0	122.8	3.6
1900.0	126.5	-0.1
2000.0	129.1	-2.7

Table 56. Expected output for VERTMIX for r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
50.0	141.8	-35.4
100.0	134.7	-28.3
150.0	130.4	-24.0
200.0	127.2	-20.8
250.0	124.6	-18.2
300.0	122.4	-16.0
350.0	120.5	-14.1
400.0	118.9	-12.5
450.0	117.6	-11.1
500.0	116.4	-10.0
550.0	115.4	-9.0
600.0	114.5	-8.1
650.0	113.7	-7.3
700.0	112.9	-6.4
750.0	112.6	-6.2
800.0	112.0	-5.6
850.0	111.4	-5.0
900.0	110.8	-4.4
950.0	110.3	-3.9
1000.0	109.8	-3.4

Table 57. Expected output for VERTSEA for r_{max} receiver range of 300 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
50.0	136.4	-14.4
100.0	129.9	-7.9
150.0	127.1	-5.1
200.0	126.7	-4.7
250.0	129.2	-7.2
300.0	135.9	-13.9
350.0	143.8	-21.8
400.0	147.6	-25.6
450.0	146.2	-24.2
500.0	144.9	-22.9
550.0	144.2	-22.3
600.0	144.0	-22.0
650.0	144.0	-22.0
700.0	144.2	-22.2
750.0	144.6	-22.6
800.0	145.0	-23.0
850.0	145.5	-23.5
900.0	145.9	-23.9
950.0	146.3	-24.3
1000.0	146.6	-24.6

Table 58. Expected output for VERTUSRD
for r_{max} receiver range of 50 km.

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
50.0	140.5	-34.1
100.0	134.0	-27.5
150.0	129.8	-23.4
200.0	126.8	-20.3
250.0	124.3	-17.9
300.0	122.2	-15.8
350.0	120.5	-14.1
400.0	119.0	-12.5
450.0	117.7	-11.2
500.0	116.5	-10.1
550.0	115.4	-9.0
600.0	114.5	-8.1
650.0	113.7	-7.2
700.0	112.9	-6.5
750.0	112.6	-6.2
800.0	112.0	-5.6
850.0	111.4	-5.0
900.0	110.8	-4.4
950.0	110.3	-3.9
1000.0	109.8	-3.4

Table 59. Expected output for WEDGE
for r_{max} receiver range of 100 km..

Height (meters)	Prop. Loss (dB)	Prop. Factor (dB)
50.0	154.5	-22.0
100.0	154.7	-22.3
150.0	155.7	-23.3
200.0	155.5	-23.0
250.0	154.0	-21.6
300.0	152.1	-19.7
350.0	150.3	-17.9
400.0	149.0	-16.6
450.0	148.1	-15.6
500.0	146.8	-14.4
550.0	144.0	-11.5
600.0	139.7	-7.3
650.0	135.4	-2.9
700.0	131.6	0.9
750.0	128.3	4.1
800.0	126.3	6.2
850.0	126.9	5.6
900.0	127.8	4.6
950.0	127.4	5.0
1000.0	130.2	2.2

4.5 CRITERIA FOR EVALUATING RESULTS

The calculated propagation loss in dB should match the numerical values in each table at each of the levels shown to within 0.1 dB (1 cB). APM rounds its output loss values to the nearest 1 cB, and hence it is possible for differences of 1 cB to exist between different implementations of APM. It is expected, however, that in most cases the values will match those in Table 24 through Table 59 exactly.

4.6 TEST PROCEDURE

1. Compile for execution, the APM CSCI, the driver program APMMAIN.F90, and the module APM_MOD.F90.
2. An input data file has been provided, as a text file, for each test case.
3. The APM CSCI is executed in a form that reads the input data file, performs the calculations, and writes the output to a text file.
4. The output file is compared to the final expected test results to determine satisfactory performance.

4.7 ASSUMPTIONS AND CONSTRAINTS

Input data elements are assumed to be constrained by the limits listed within Tables 1 through 4 of the Software Requirements Specification (Ref. 5).

5. REQUIREMENTS TRACEABILITY

The provided driver program that accesses the APM CSCI will create an output file for each test case. The output file will have the same prefix name as the input file. The extension is “.OUT”. This output file contains height in meters and corresponding propagation loss in dB that should correspond to the entries in Table 24 through Table 59 for each test case.

The provided program APMMAIN.FOR, when compiled with the APM CSCI, will read the provided input files containing all necessary information for each test case. Each input file is named for each test case, with a “.IN” extension.

6. NOTES

Table 60 is a glossary of acronyms and abbreviations used within this document.

Table 60. Acronyms and abbreviations.

Term	Definition
abs_{hum}	Surface absolute humidity (g/m^3)
ant_{ht}	Antenna height
APM	Advanced Propagation Model
μ_{bw}	Antenna vertical beam width (degrees)
cB	centibel
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
dB	Decibel
$dielec$	2-dimensional array of relative permittivity and conductivity
μ_o	antenna elevation angle (degrees)
EM	Electromagnetic
FORTTRAN	Formula Translation
f_{MHz}	EM system frequency (MHz)
γ_a	Surface specific attenuation rate (dB/km)
$hfang$	User-defined height-finder power reduction angle array (deg)
$hfac$	User-defined power reduction factor array
h_{max}	Maximum height output for a particular application of APM.
h_{min}	Minimum height output for a particular application of APM.
$hmsl$	Refractivity profile height array
i_{extra}	Extrapolation flag for refractivity profiles entered below mean sea level
i_{gr}	Number of ground composition types for particular application of APM
$igrnd$	Ground composition type array
i_{pat}	Antenna pattern
i_{pol}	Antenna polarization
i_{tp}	Number of terrain points for particular application of APM
$lerr6$	Controlling logical flag for error 6
$lerr12$	Controlling logical flag for error 12
$lvlp$	Number of levels in refractivity profiles for particular application of APM

Table 60. Acronyms and abbreviations. (Continued)

Term	Definition
km	kilometers
m	meters
N/A	Not applicable
n_{fac}	Number of power reduction factors and cut-back angles for user-defined height-finder radar
n_{prof}	Number of refractivity profiles for particular application of APM
n_{rout}	Number of range output points for a particular application of APM.
n_w	Number of wind speeds
n_{zout}	Number of height output points for a particular application of APM.
PE_{flag}	Logical flag indicating PE-only mode
$refmsl$	Refractivity profile M-unit array
$rgrnd$	Ground composition type range array
r_{max}	Maximum range output for a particular application of APM.
r_{mult}	PE range step multiplier
$rngprof$	Refractivity profile range array
$rngwind$	Range array of wind speeds
t_{air}	Surface air temperature (°C)
$terx$	Terrain profile range array
$tery$	Terrain profile height array
th_{max}	Visible portion of maximum PE propagation angle
T_{ropo}	Logical flag to include troposcatter calculations
$wind$	Wind speed array (m/s)
NITES	Naval Integrated Tactical Environmental Subsystem

7. SAMPLE PROGRAM LISTING

The sample driver program APMMAIN.FOR, which exercises the APM CSCI, is provided below.

```
!***** APMMAIN DRIVER PROGRAM FOR APM Ver 1.3.1 *****

! This is a sample driver program for APM routines APMINIT, APMSTEP,
! XOINIT, and XOSTEP. All numeric parameters passed to APMINIT and
! APMSTEP must be in metric units. All input arrays are dynamically
! allocated and are dimensioned with variable sizes.

program apmmain

use apm_mod

implicit integer(kind=4) (i-n)
implicit real(kind=8) (a-h, o-z)

!MPFL must be declared an INTEGER*2 allocatable array.
!ITLOSS is a dummy array and will be used to store entire loss grid.
!ITPFAC is a dummy array and will be used to store entire propagation factor grid.

!NOTE: Propagation factor is output as 20*LOG10(F).

integer(kind=2), allocatable :: mpfl(:,,:), itloss(:,,:), itpfac(:,,:)

character filein*20, fileout*24, answer*1

10 continue

write(*,'(a\)' )' Name of input file? '
read(*, '(a)' ) filein

open(14, file=filein)

!*****READ CALC INFO*****

read( 14, * ) lerr6
read( 14, * ) lerr12

read( 14, * ) peflag !Perform field calcs using PE model only?
read( 14, * ) thmax !Maximum PE calculation angle in degrees (used only if
!PEFLAG = .true.
read( 14, * ) rmult !PE range step multiplier (used only if PEFLAG = .true.)
read( 14, * ) tropo !Troposcatter flag: .false.=no troposcatter, .true.=troposcatter

!*****READ SYSTEM INFO*****

read( 14, * ) freq !Frequency in MHz.
read( 14, * ) antht !antenna height.
read( 14, * ) ipat !antenna type
read( 14, * ) ipol !antenna polarization.

!This value is ignored for Omni antenna, otherwise, the value must be
!entered in degrees.

read( 14, * ) bwidth

!This value is ignored for Omni antenna, otherwise, the value must be
!entered in degrees.

read( 14, * ) elev

!If using specific height-finder antenna, this variable contains a non-zero
!value corresponding to the # of cut-back angles and cut-back factors.

read( 14, * ) nfacs

! If using specific height-finder antenna, then must specify values for HFANG() and
! HFFAC arrays. Height-finder cut-back angles HFANG() must be in degrees.
```

```

if( nfacs .gt. 0 ) then
  IF( ALLOCATED( hfang ) ) DEALLOCATE( hfang, stat=ierror )
  ALLOCATE( hfang(nfacs), stat=ierror )
  if( ierror .ne. 0 ) then
    write(*,*)'*****ERROR IN HFANG ALLOCATION*****'
    stop
  end if
  hfang = 0.

  IF( ALLOCATED( hffac ) ) DEALLOCATE( hffac, stat=ierror )
  ALLOCATE( hffac(nfacs), stat=ierror )
  if( ierror .ne. 0 ) then
    write(*,*)'*****ERROR IN HFFAC ALLOCATION*****'
    stop
  end if
  hffac = 0.

  do i = 1, nfacs
    read( 14, * ) hfang(i), hffac(i)
  end do
end if

!*****READ GENERIC INPUT INFO*****

read( 14, * ) hmin      !Minimum height in m
read( 14, * ) hmax      !Maximum output height in m
read( 14, * ) rkm       !Maximum output range in km
rmax = rkm * 1.d3       !Convert to m and initialize RMAX for input to APM.
read( 14, * ) nzout     !Number of output height points.
read( 14, * ) nrout     !Number of output range points.

!*****READ METEOROLOGICAL INFO*****

read( 14, * ) iextra    !Extrapolation flag:
                        !0=extrapolate using standard gradient,
                        !1=extrapolate using gradient from first 2 levels.
read( 14, * ) abshum    !Surface absolute humidity in g/m**3
read( 14, * ) tair      !Surface air temperature in degrees C
read( 14, * ) gammaa    !Gaseous absorption attenuation rate in dB/km

read( 14, * ) nw        !Number of wind speeds specified.

if( nw .gt. 0 ) then    !If wind speeds specified, allocate memory.

  IF( ALLOCATED( RNGWIND ) ) DEALLOCATE( RNGWIND )
  ALLOCATE( RNGWIND(nw) )
  RNGWIND = 0.

  IF( ALLOCATED( WIND ) ) DEALLOCATE( WIND )
  ALLOCATE( WIND(nw) )
  WIND = 0.

!Read wind speeds and ranges.

  do i = 1, nw
    read( 14, * ) wind(i), rngwind(i)  !Wind speed in m/s and range in km at
  end do                                !which to apply specified wind speed.
  rngwind = 1.d3 * rngwind             !Convert RNGWIND from km to m.
end if

read( 14, * ) nprof     !Number of refractivity profiles
read( 14, * ) lvlp      !Number of levels in refractivity profiles.

! Allocate and initialize height/refractivity and range arrays.

IF( ALLOCATED( HMSL ) ) DEALLOCATE( HMSL, stat=ierror )
ALLOCATE( HMSL(0:LVL, NPROF), stat=ierror )
if( ierror .ne. 0 ) then
  write(*,*)'*****ERROR IN HMSL ALLOCATION*****'
  stop
end if
HMSL = 0.

IF( ALLOCATED( REFMSL ) ) DEALLOCATE( REFMSL, stat=ierror )

```

```

ALLOCATE( REFMSL(0:LVLP, NPROF), stat=ierror )
if( ierror .ne. 0 ) then
    write(*,*)'*****ERROR IN REFMSL ALLOCATION*****'
    stop
end if
REFMSL = 0.

IF( ALLOCATED( RNGPROF ) ) DEALLOCATE( RNGPROF, stat=ierror )
ALLOCATE( RNGPROF(NPROF), stat=ierror )
if( ierror .ne. 0 ) then
    write(*,*)'*****ERROR IN RNGPROF ALLOCATION*****'
    stop
end if
RNGPROF = 0.

do i = 1, nprof
    read( 14, * ) rngp                !Range of profile in km
    rngprof(i) = rngp * 1.d3          !Convert profile range from km to m
    do j = 0, lvlp-1
        read( 14, * ) hmsl(j,i), refmsl(j,i) !Height/refractivity levels
    end do
end do

!*****READ TERRAIN INFO*****

read( 14, * ) igr                    !Number of ground composition types
if( igr .gt. 0 ) then

    IF( ALLOCATED( DIELEC ) ) DEALLOCATE( DIELEC, stat=ierror )
    ALLOCATE( DIELEC(2, IGR), stat=ierror )
    if( ierror .ne. 0 ) then
        write(*,*)'*****ERROR IN DIELEC ALLOCATION*****'
        stop
    end if
    DIELEC = 0.

    IF( ALLOCATED( IGRND ) ) DEALLOCATE( IGRND, stat=ierror )
    ALLOCATE( IGRND(IGR), stat=ierror )
    if( ierror .ne. 0 ) then
        write(*,*)'*****ERROR IN IGRND ALLOCATION*****'
        stop
    end if
    IGRND = 0.

    IF( ALLOCATED( RGRND ) ) DEALLOCATE( RGRND, stat=ierror )
    ALLOCATE( RGRND(IGR), stat=ierror )
    if( ierror .ne. 0 ) then
        write(*,*)'*****ERROR IN RGRND ALLOCATION*****'
        stop
    end if
    RGRND = 0.

    ! Read ranges at which ground types apply, ground composition types, and dielectric
    ! constants. If IGRND(i) = 7, then must specify non-zero values for DIELEC(), otherwise
    ! set to 0. Ranges of ground types are read in km.

    do i = 1, igr
        read( 14, * ) rground, igrnd(i), (dielec(j,i),j=1,2)
        rgrnd(i) = rground * 1.d3
    end do

end if

read( 14, * ) itp                    !Number of terrain range/height points
if( itp .gt. 1 ) then ! Valid terrain profile must contain at least two
    ! height/range points.

    IF( ALLOCATED( TERX ) ) DEALLOCATE( TERX, stat=ierror )
    ALLOCATE( TERX(ITP), stat=ierror )
    if( ierror .ne. 0 ) then
        write(*,*)'*****ERROR IN TERX ALLOCATION*****'
        stop
    end if
    TERX = 0.

```



```

        IF( ALLOCATED( TERY ) ) DEALLOCATE( TERY, stat=ierror )
        ALLOCATE( TERY(ITP), stat=ierror )
        if( ierror .ne. 0 ) then
            write(*,*)'*****ERROR IN TERY ALLOCATION*****'
            stop
        end if
        TERY = 0.

        do i = 1, itp
            read( 14, * ) terrain_x, tery(i)
            terx(i) = terrain_x * 1.d3
        end do

    end if
    close(14)

!Allocate and initialize MPFL() and dummy arrays.

if( allocated( mpfl ) ) deallocate( mpfl, stat=ierror )
allocate( mpfl(2,0:nzout), stat = ierror )
if( ierror .ne. 0 ) then
    write(*,*)'*****ERROR IN MPFL ALLOCATION*****'
    stop
end if
mpfl = 0

if( allocated( itloss ) ) deallocate( itloss, stat=ierror )
allocate( itloss(0:nzout,nrout), stat=ierror )
if( ierror .ne. 0 ) then
    write(*,*)'*****ERROR IN ITLOSS ALLOCATION*****'
    stop
end if
itloss = 0

if( allocated( itpfac ) ) deallocate( itpfac, stat=ierror )
allocate( itpfac(0:nzout,nrout), stat=ierror )
if( ierror .ne. 0 ) then
    write(*,*)'*****ERROR IN ITPFAC ALLOCATION*****'
    stop
end if
itpfac = 0

! Write all inputs that create the resulting output propagation loss values as
! part of log file.

ip = index( filein, '.' )
if( ip .gt. 0 ) then
    fileout = filein(1:ip-1)//'.out'
else
    ic = len_trim( filein )
    fileout = filein(1:ic)//'.out'
end if

open( 15, file=fileout )

write( 15, * )'****Input Log APM 1.3.1****'
write( 15, * )'lerr6 = ', lerr6
write( 15, * )'lerr12 = ', lerr12
write( 15, * )'PE-only flag = ', peflag
write( 15, '(a,f10.2)' )'Maximum PE angle (deg) = ', thmax
write( 15, '(a,f10.2)' )'PE range step multiplier = ', rmult
write( 15, * )'Troposcatter calcs? = ', tropo
write( 15, '(a,f10.2)' )'Frequency (MHz) = ', freq
write( 15, '(a,f10.3)' )'Antenna height (m) = ', antht
write( 15, * )'Antenna type = ', ipat
write( 15, * )'Polarization = ', ipol
write( 15, '(a,f10.3)' )'Beamwidth (deg) = ', bwidth
write( 15, '(a,f10.3)' )'Elevation angle (deg) = ', elev
write( 15, * )'Number of cut-back angles and factors = ', nfacs
if( nfacs .gt. 0 ) then
    write( 15, * )'Cut-back angles (deg) = ', ( sngl(hfang(i)), i=1, nfacs )
    write( 15, * )'Cut-back factors = ', ( sngl(hffac(i)), i=1, nfacs )
end if
write( 15, '(a,f10.2)' )'Minimum output height (m) = ', hmin
write( 15, '(a,f10.2)' )'Maximum output height (m) = ', hmax
write( 15, '(a,f10.2)' )'Maximum output range (km) = ', rkm

```

```

write( 15, * )'Number of output height points = ', nzout
write( 15, * )'Number of output range points = ', nrout
write( 15, * )'Extrapolation flag = ', iextra
write( 15, '(a,f10.3)' )'Surface absolute humidity in g/m**3 = ', abshum
write( 15, '(a,f10.3)' )'Surface air temperature in degrees C = ', tair
write( 15, '(a,f10.3)' )'Gaseous absorption attenuation rate in dB/km = ', gammaa
write( 15, * )'Number of wind speeds specified = ', nw

if( nw .gt. 0 ) then
  write( 15, '(a,f10.2)' )'Wind speeds (m/s) = ', (wind(i), i=1,nw)
  write( 15, '(a,f10.2)' )'Range at each wind speed (km) = ', ( rngwind(i)*1.d-3, i=1,nw
)
end if

write( 15, * )'Number of refractivity profiles = ', nprof
write( 15, * )'Number of levels in refractivity profiles = ', lvlp

do j = 1, nprof
  write( 15, '(a,i2,a,f10.1)' )'Range of profile ', j, ' in km = ',rngprof(j)*1.d-3
  write(15,*)'Height (m)', '    M-unit for Profile', j
  do i = 0, lvlp-1
    write(15,'(2f15.3)') hmsl(i,j), refmsl(i,j)
  end do
end do

write( 15, * )'Number of ground composition types = ', igr
write( 15, * )'Range(m) of ground type          Ground types          Dielec(perm.,cond.)'
do i = 1, igr
  write( 15, '(f15.2,20x,i1,7x,2(f15.2))' ) rgrnd(i), igrnd(i), (dielec(j,i), j=1,2)
end do

write( 15, * )'Number of terrain range/height points = ', itp
if( itp .gt. 1 ) then
  write(15,*)'Range (km)', '    Height (m)'
  do i = 1, itp
    write( 15, '(2f15.3)' ) terx(i)*1.d-3, tery(i)
  end do
end if

hmin_bef = hmin
hmax_bef = hmax

alimv = 0.d0 ! ***MAKE SURE THIS VARIABLE IS INITIALIZED TO ZERO BEFORE ANY CALLS TO
APMINIT.***
! ***THIS IS FOR SPAWAR USE ONLY.***

! Variables in CAPS are returned.

call apminit( IXOSTP, IERROR )

if( ierror .ne. 0 ) then
  write(*,*)'***** ERROR IN APMINIT *****'
  write(*,*)'***** IERROR = ', ierror,' *****'
  stop
end if

! Notify user that HMIN or HMAX has been changed on return from APMINIT.
! The calculation height (HMAX-HMIN) must be at least 100 m.

hmin_aft = hmin
if( dabsh(hmin_bef - hmin_aft) .gt. 1.d-3 ) then
  write(15,*)
  write(15,*)' *****WARNING*****'
  write(15,*)'HMIN has been adjusted to ', hmin, 'meters'
  write(15,*)' *****'
end if

hmax_aft = hmax
if( dabsh(hmax_bef - hmax_aft) .gt. 1.d-3 ) then
  write(15,*)
  write(15,*)' *****WARNING*****'
  write(15,*)'HMAX has been adjusted to ', hmax, 'meters'
  write(15,*)' *****'
end if

do istp = 1, nrout

```

```

! JSTART = start of valid loss points, JEND = end of valid loss
! points. If at a range where extended optics will be applied, then
! JEND will be the index at top of PE region in MPFL().

call apmstep( istp, ROUT, MPFL, JSTART, JEND )

write(*,*)'range in km = ', rout*1.d-3 !Output to screen

! Store loss and propagation factor points in 2-dim. grid for later output to file.

itloss( 0:nzout, istp ) = mpfl( 1, 0:nzout ) !prop loss
itpfac( 0:nzout, istp ) = mpfl( 2, 0:nzout ) !prop factor
end do

! Initialize variables to be used in XO model.

call xoinit( ixostp, jend, JXSTART, IERROR )
if( ierror .gt. 0 ) then
    write(*,*)'*****ERROR IN XOINIT*****'
    stop
end if

! If extended optics model needs to be used, then call.

if( ixostp .gt. 0 ) then

    do istp = ixostp, nrout

        call xostep( istp, ROUT, MPFL, jxstart, JXEND )
        write(*,*)'range in km (XO region) = ', rout*1.d-3 !Output to screen
        itloss( jxstart:jxend, istp ) = mpfl( 1, jxstart:jxend )
        itpfac( jxstart:jxend, istp ) = mpfl( 2, jxstart:jxend )

    end do

end if

! NOTE: If V pol is specified, then there can be NZOUT + 1 valid loss points
! at each range, where the extra point is stored in MPFL(0). However,
! only NZOUT loss values, from MPFL(1) to MPFL(NZOUT), will be written to the
! output file. In order to automatically output NZOUT for H pol cases or
! NZOUT+1 points for V pol cases, modify the "do loop" below as follows:
!     do j = 1, nrout
!         .....write statements identical....
!         do k = io, nzout
!             .....
!         end do
!     end do
! IO is a common variable set within APMINIT that equals 0 or 1 depending on the
! polarization used.

! Now store all loss values in output file FILEOUT.
! Recall that MPFL is the propagation loss/factor in centibels, i.e.,
! MPFL() = NINT( propagation loss/factor in dB * 10. ).

! Output height increment DZOUT, as determined in APMINIT, is computed as
! DZOUT = (HMAX-HMIN) / float( NZOUT )

! Output range increment DROUT, as determined in APMINIT, is computed as
! DROUT = RMAX / float( NROUT )

! Loop for writing propagation loss & factor vs. height for a specified range.

write( 15, * )
write( 15, * )'*****Output Loss and Prop. Factor Values*****'

do j = 1, nrout
    write(15,*)
    write(15,(a,f10.2))'range in km = ', real(j,8)*drou*1.d-3
    write(15,*)
    write(15,*)'Height(m)    Loss(dB)    PFac(dB) '
    do k = 1, nzout
        if( itloss(k,j) .eq. -1000 ) then
            ploss = -1000.
            pfac = -1000.
        end if
    end do
end do

```

```

        elseif( itloss(k,j) .eq. -999 ) then
            ploss = -999.
            pfac = -999.
        else
            ploss = itloss(k,j)*.1
            pfac = itpfac(k,j)*.1
        end if
        write(15,'(3f10.2)') hmin + real(k,8)*dzout, ploss, pfac
    end do
end do

! Loop for writing propagation loss & factor vs. range for a specified height.

!do k = io, nzout
!    write(15,*)
!    write(15,'(a,f10.2)') 'Height in m = ', real(k,8)*dzout
!    write(15,*)
!    write(15,*) 'Range (km)          Loss (dB)    Prop. factor(dB)  '
!    do j = 1, nrout
!        if( itloss(k,j) .eq. -1000 ) then
!            ploss = -1000.
!            pfac = -1000.
!        elseif( itloss(k,j) .eq. -999 ) then
!            ploss = -999.
!            pfac = -999.
!        else
!            ploss = itloss(k,j)*.1
!            pfac = itpfac(k,j)*.1
!        end if
!        write(15,'(3f10.2)') real(j,8)*drout*1.d-3, ploss, pfac
!    end do
!end do

close(15)

!Deallocate all allocated arrays in main driver program before exiting.

if( allocated( hfang ) ) deallocate( hfang, stat=ierror )
if( ierror .ne. 0 ) then
    write( *, *) '*****ERROR IN HFANG DEALLOCATION*****'
    stop
end if

if( allocated( hffac ) ) deallocate( hffac, stat=ierror )
if( ierror .ne. 0 ) then
    write( *, *) '*****ERROR IN HFFAC DEALLOCATION*****'
    stop
end if

if( allocated( terx ) ) deallocate( terx, stat=ierror )
if( ierror .ne. 0 ) then
    write( *, *) '*****ERROR IN TERX DEALLOCATION*****'
    stop
end if

if( allocated( tery ) ) deallocate( tery, stat=ierror )
if( ierror .ne. 0 ) then
    write( *, *) '*****ERROR IN TERY DEALLOCATION*****'
    stop
end if

deallocate( mpfl, itloss, itpfac )

write(*,'(a\)\') ' Input another file? (y or n)'
read(*, '(a)\') answer
if(( answer .eq. 'y' ) .or. ( answer .eq. 'Y')) goto 10

end

```

8. INPUT FILE LISTINGS FOR TEST CASES

Each test case, when using the sample driver program APMMAIN.F90, shall consist of an input file (*TestName.IN*) and an output file (*TestName.OUT*). The input file's contents are listed in sections 8.1 through 8.28. The output file's contents, consisting of couplets of height in meters and propagation loss and propagation factor in dB, are listed in Table 24 through Table 59.

8.1 ABSORB.IN

```
.true.      : LERR6 error flag
.true.      : LERR12 error flag
.false.     : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.          : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.          : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.     : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
20000.      : Frequency in MHz
25.         : Antenna height in m
1           : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0           : Polarization (0=HOR, 1=VER)
5.          : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.          : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0           : Number of cut-back angles and factors (used for specific height-finder antenna)
0.          : Minimum output height in m
200.        : Maximum output height in m
50.         : Maximum output range in km
20          : Number of output height points
1           : Number of output range points
0           : Extrapolation flag
0.          : Surface absolute humidity in g/m3
0.          : Surface air temperature in degrees
.146        : Gaseous absorption attenuation rate in dB/km
0           : Number of wind speeds/ranges specified
1           : Number of refractivity profiles
2           : Number of levels in refractivity profiles
0.          : Range of first refractivity profile in km
0.          : 350.      : Height & M-unit value of ref. profile 1, level 1
1000.       : 468.      : Height & M-unit value of ref. profile 1, level 2
1           : Number of ground composition types
0., 0, 0., 0. : Range(km), ground type (integer), permittivity, conductivity
0           : Number of terrain range/height points
```

8.2 AIRBORNE.IN

```
.true.      : LERR6 error flag
.true.      : LERR12 error flag
.false.     : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.          : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.          : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.     : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
900.        : Frequency in MHz
2500.       : Antenna height in m
1           : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0           : Polarization (0=HOR, 1=VER)
0.          : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.          : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0           : Number of cut-back angles and factors (used for specific height-finder antenna)
0.          : Minimum output height in m
5000.       : Maximum output height in m
250.        : Maximum output range in km
20          : Number of output height points
1           : Number of output range points
0           : Extrapolation flag
```

```

0.      : Surface absolute humidity in g/m3
0.      : Surface air temperature in degrees
0.      : Gaseous absorption attenuation rate in dB/km
0       : Number of wind speeds/ranges specified
1       : Number of refractivity profiles
5       : Number of levels in refractivity profiles
0.      : Range of first refractivity profile in km
0.      209.2   : Height & M-unit value of ref. profile 1, level 1
1100.   339.    : Height & M-unit value of ref. profile 1, level 2
1500.   386.2   : Height & M-unit value of ref. profile 1, level 3
1625.   361.5   : Height & M-unit value of ref. profile 1, level 4
2100    417.55  : Height & M-unit value of ref. profile 1, level 5
1       : Number of ground composition types
0., 0, 0., 0.   : Range(km), ground type (integer), permittivity, conductivity
0       : Number of terrain range/height points

```

8.3 BLOCK.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.false.  : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.       : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
1000.    : Frequency in MHz
101.     : Antenna height in m
1        : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
1        : Polarization (0=HOR, 1=VER)
0.       : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.       : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0        : Number of cut-back angles and factors (used for specific height-finder antenna)
0.       : Minimum output height in m
400.     : Maximum output height in m
60.      : Maximum output range in km
20       : Number of output height points
1        : Number of output range points
0        : Extrapolation flag
7.5      : Surface absolute humidity in g/m3
0.       : Surface air temperature in degrees
0.       : Gaseous absorption attenuation rate in dB/km
0        : Number of wind speeds/ranges specified
1        : Number of refractivity profiles
2        : Number of levels in refractivity profiles
0.       : Range of first refractivity profile in km
0        350    : Height & M-unit value of ref. profile 1, level 1
1000    468    : Height & M-unit value of ref. profile 1, level 2
1       : Number of ground composition types
0., 7, 7.5, 0.01 : Range(km), ground type (integer), permittivity, conductivity
6       : Number of terrain range/height points
0.      1      : Range(km) & height of terrain point 1
10.0   1       : Range(km) & height of terrain point 2
10.0   200
40.0   200
40.0   1
60.0   1

```

8.4 COSEC2.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.false.  : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.       : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
1000.    : Frequency in MHz
25.      : Antenna height in m
4        : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0        : Polarization (0=HOR, 1=VER)
1.       : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)

```

```

0.      : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0       : Number of cut-back angles and factors (used for specific height-finder antenna)
0.      : Minimum output height in m
2000.   : Maximum output height in m
50.     : Maximum output range in km
20      : Number of output height points
1       : Number of output range points
0       : Extrapolation flag
0.      : Surface absolute humidity in g/m3
0.      : Surface air temperature in degrees
0.      : Gaseous absorption attenuation rate in dB/km
0       : Number of wind speeds/ranges specified
1       : Number of refractivity profiles
2       : Number of levels in refractivity profiles
0.      : Range of first refractivity profiles in km
0.      350.      : Height & M-unit value of ref. profile 1, level 1
1000.   468.     : Height & M-unit value of ref. profile 1, level 2
1       : Number of ground composition types
0., 0, 0., 0.   : Range(km), ground type (integer), permittivity, conductivity
0       : Number of terrain range/height points

```

8.5 EDUCT.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.false.  : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.       : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
10000.   : Frequency in MHz
15.      : Antenna height in m
2        : Antenna type (1=OMNI, 2=GAUSS, 3=SINC(X), 4=COSEC2, 5=HTFIND, 6=USRHTFIND, 7=USRDEF)
0        : Polarization (0=HOR, 1=VER)
5.       : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.       : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0        : Number of cut-back angles and factors (used for specific height-finder antenna)
0.       : Minimum output height in m
200.     : Maximum output height in m
50.      : Maximum output range in km
20       : Number of output height points
1        : Number of output range points
0        : Extrapolation flag
0.       : Surface absolute humidity in g/m3
0.       : Surface air temperature in degrees
0.       : Gaseous absorption attenuation rate in dB/km
0        : Number of wind speeds/ranges specified
1        : Number of refractivity profiles
21       : Number of levels in refractivity profiles
0.       : Range of first refractivity profiles in km
0.      339.     : Height & M-unit value of ref. profile 1, level 1
.040     335.10  : Height & M-unit value of ref. profile 1, level 2
.1       333.66  : Height & M-unit value of ref. profile 1, level 3
.2       332.6   : Height & M-unit value of ref. profile 1, level 4
.398     331.54  : Height & M-unit value of ref. profile 1, level 5
.794     330.51  : Height & M-unit value of ref. profile 1, level 6
1.585    329.53  : Height & M-unit value of ref. profile 1, level 7
3.162    328.65  : Height & M-unit value of ref. profile 1, level 8
6.310    327.96  : Height & M-unit value of ref. profile 1, level 9
12.589   327.68  : Height & M-unit value of ref. profile 1, level 10
14.      327.67  : Height & M-unit value of ref. profile 1, level 11
25.119   328.13  : Height & M-unit value of ref. profile 1, level 12
39.811   329.25  : Height & M-unit value of ref. profile 1, level 13
50.119   330.18  : Height & M-unit value of ref. profile 1, level 14
63.096   331.44  : Height & M-unit value of ref. profile 1, level 15
79.433   333.12  : Height & M-unit value of ref. profile 1, level 16
100.     335.33  : Height & M-unit value of ref. profile 1, level 17
125.893  338.2   : Height & M-unit value of ref. profile 1, level 18
158.489  341.92  : Height & M-unit value of ref. profile 1, level 19
199.526  346.69  : Height & M-unit value of ref. profile 1, level 20
209.526  347.87  : Height & M-unit value of ref. profile 1, level 21
1        : Number of ground composition types
0., 0, 0., 0.   : Range(km), ground type (integer), permittivity, conductivity

```

0 : Number of terrain range/height points

8.6 EDUCTRF.IN

```
.true.      : LERR6 error flag
.true.      : LERR12 error flag
.false.     : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.          : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.          : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.     : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
10000.      : Frequency in MHz
15.         : Antenna height in m
2           : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0           : Polarization (0=HOR, 1=VER)
5.          : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.          : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0           : Number of cut-back angles and factors (used for specific height-finder antenna)
0.          : Minimum output height in m
200.        : Maximum output height in m
100.        : Maximum output range in km
20          : Number of output height points
1           : Number of output range points
0           : Extrapolation flag
0.          : Surface absolute humidity in g/m3
0.          : Surface air temperature in degrees
0.          : Gaseous absorption attenuation rate in dB/km
1           : Number of wind speeds/ranges specified
10., 0.     : Wind speed (m/s), Range(km)
1           : Number of refractivity profiles
21          : Number of levels in refractivity profiles
0.          : Range of first refractivity profiles in km
0.          : 339. : Height & M-unit value of ref. profile 1, level 1
.040        : 335.10 : Height & M-unit value of ref. profile 1, level 2
.1          : 333.66 : Height & M-unit value of ref. profile 1, level 3
.2          : 332.6 : Height & M-unit value of ref. profile 1, level 4
.398        : 331.54 : Height & M-unit value of ref. profile 1, level 5
.794        : 330.51 : Height & M-unit value of ref. profile 1, level 6
1.585       : 329.53 : Height & M-unit value of ref. profile 1, level 7
3.162       : 328.65 : Height & M-unit value of ref. profile 1, level 8
6.310       : 327.96 : Height & M-unit value of ref. profile 1, level 9
12.589      : 327.68 : Height & M-unit value of ref. profile 1, level 10
14.         : 327.67 : Height & M-unit value of ref. profile 1, level 11
25.119      : 328.13 : Height & M-unit value of ref. profile 1, level 12
39.811      : 329.25 : Height & M-unit value of ref. profile 1, level 13
50.119      : 330.18 : Height & M-unit value of ref. profile 1, level 14
63.096      : 331.44 : Height & M-unit value of ref. profile 1, level 15
79.433      : 333.12 : Height & M-unit value of ref. profile 1, level 16
100.        : 335.33 : Height & M-unit value of ref. profile 1, level 17
125.893     : 338.2 : Height & M-unit value of ref. profile 1, level 18
158.489     : 341.92 : Height & M-unit value of ref. profile 1, level 19
199.526     : 346.69 : Height & M-unit value of ref. profile 1, level 20
209.526     : 347.87 : Height & M-unit value of ref. profile 1, level 21
1           : Number of ground composition types
0., 0, 0., 0. : Range(km), ground type (integer), permittivity, conductivity
0           : Number of terrain range/height points
```

8.7 FLTA50.IN

```
.true.      : LERR6 error flag
.true.      : LERR12 error flag
.false.     : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.          : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.          : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.     : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
1000.       : Frequency in MHz
50.         : Antenna height in m
1           : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
1           : Polarization (0=HOR, 1=VER)
5.          : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
```



```

0.      : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0      : Number of cut-back angles and factors (used for specific height-finder antenna)
0.      : Minimum output height in m
100.    : Maximum output height in m
50.     : Maximum output range in km
20      : Number of output height points
1       : Number of output range points
0       : Extrapolation flag
0.      : Surface absolute humidity in g/m3
0.      : Surface air temperature in degrees
0.      : Gaseous absorption attenuation rate in dB/km
0       : Number of wind speeds/ranges specified
1       : Number of refractivity profiles
2       : Number of levels in refractivity profiles
0.      : Range of first refractivity profiles in km
0.      350.    : Height & M-unit value of ref. profile 1, level 1
1000.   468.    : Height & M-unit value of ref. profile 1, level 2
1       : Number of ground composition types
0., 7, 7., 0.01 : Range(km), ground type (integer), permittivity, conductivity
2       : Number of terrain range/height points
0.      10
50.0    10

```

8.8 GASABS.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.false.  : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.       : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
20000.   : Frequency in MHz
25.      : Antenna height in m
1        : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0        : Polarization (0=HOR, 1=VER)
5.       : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.       : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0        : Number of cut-back angles and factors (used for specific height-finder antenna)
0.       : Minimum output height in m
200.     : Maximum output height in m
50.      : Maximum output range in km
20       : Number of output height points
1        : Number of output range points
0        : Extrapolation flag
10.      : Surface absolute humidity in g/m3
25.      : Surface air temperature in degrees
0.       : Gaseous absorption attenuation rate in dB/km
0        : Number of wind speeds/ranges specified
1        : Number of refractivity profiles
2        : Number of levels in refractivity profiles
0.       : Range of first refractivity profiles in km
0.      350.    : Height & M-unit value of ref. profile 1, level 1
1000.   468.    : Height & M-unit value of ref. profile 1, level 2
1        : Number of ground composition types
0., 0, 0., 0.  : Range(km), ground type (integer), permittivity, conductivity
0        : Number of terrain range/height points

```

8.9 GAUSS.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.false.  : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.       : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
1000.    : Frequency in MHz
25.      : Antenna height in m
2        : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0        : Polarization (0=HOR, 1=VER)

```

```

1.      : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.      : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0       : Number of cut-back angles and factors (used for specific height-finder antenna)
0.      : Minimum output height in m
2000.   : Maximum output height in m
50.     : Maximum output range in km
20      : Number of output height points
1       : Number of output range points
0       : Extrapolation flag
0.      : Surface absolute humidity in g/m3
0.      : Surface air temperature in degrees
0.      : Gaseous absorption attenuation rate in dB/km
0       : Number of wind speeds/ranges specified
1       : Number of refractivity profiles
2       : Number of levels in refractivity profiles
0.      : Range of first refractivity profiles in km
0.      350.   : Height & M-unit value of ref. profile 1, level 1
1000.   468.   : Height & M-unit value of ref. profile 1, level 2
1       : Number of ground composition types
0., 0, 0., 0. : Range(km), ground type (integer), permittivity, conductivity
0       : Number of terrain range/height points

```

8.10 HIBW.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.false.  : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.       : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
1000.    : Frequency in MHz
25.      : Antenna height in m
3        : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0        : Polarization (0=HOR, 1=VER)
45.      : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.       : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0        : Number of cut-back angles and factors (used for specific height-finder antenna)
0.       : Minimum output height in m
2000.    : Maximum output height in m
50.      : Maximum output range in km
20       : Number of output height points
1        : Number of output range points
0        : Extrapolation flag
0.       : Surface absolute humidity in g/m3
0.       : Surface air temperature in degrees
0.       : Gaseous absorption attenuation rate in dB/km
0        : Number of wind speeds/ranges specified
1        : Number of refractivity profiles
2        : Number of levels in refractivity profiles
0.       : Range of first refractivity profiles in km
0.      350.   : Height & M-unit value of ref. profile 1, level 1
1000.   468.   : Height & M-unit value of ref. profile 1, level 2
1        : Number of ground composition types
0., 0, 0., 0. : Range(km), ground type (integer), permittivity, conductivity
0        : Number of terrain range/height points

```

8.11 HIEL.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.false.  : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.       : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
1000.    : Frequency in MHz
25.      : Antenna height in m
2        : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0        : Polarization (0=HOR, 1=VER)

```

```

1.      : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
10.     : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0       : Number of cut-back angles and factors (used for specific height-finder antenna)
0.      : Minimum output height in m
20000. : Maximum output height in m
50.     : Maximum output range in km
20      : Number of output height points
1       : Number of output range points
0       : Extrapolation flag
0.      : Surface absolute humidity in g/m3
0.      : Surface air temperature in degrees
0.      : Gaseous absorption attenuation rate in dB/km
0       : Number of wind speeds/ranges specified
1       : Number of refractivity profiles
2       : Number of levels in refractivity profiles
0.      : Range of first refractivity profiles in kkm
0.      350.      : Height & M-unit value of ref. profile 1, level 1
1000.   468.     : Height & M-unit value of ref. profile 1, level 2
1       : Number of ground composition types
0., 0, 0., 0.   : Range(km), ground type (integer), permittivity, conductivity
0       : Number of terrain range/height points

```

8.12 HIFREQ.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.false.  : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.       : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
20000.   : Frequency in MHz
25.      : Antenna height in m
1        : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0        : Polarization (0=HOR, 1=VER)
0.       : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.       : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0       : Number of cut-back angles and factors (used for specific height-finder antenna)
0.      : Minimum output height in m
200.     : Maximum output height in m
50.      : Maximum output range in km
20       : Number of output height points
1        : Number of output range points
0        : Extrapolation flag
0.      : Surface absolute humidity in g/m3
0.      : Surface air temperature in degrees
0.      : Gaseous absorption attenuation rate in dB/km
0       : Number of wind speeds/ranges specified
1       : Number of refractivity profiles
2       : Number of levels in refractivity profiles
0.      : Range of first refractivity profiles in km
0.      350.     : Height & M-unit value of ref. profile 1, level 1
1000.   468.    : Height & M-unit value of ref. profile 1, level 2
1       : Number of ground composition types
0., 0, 0., 0.   : Range(km), ground type (integer), permittivity, conductivity
0       : Number of terrain range/height points

```

8.13 HITRAN.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.false.  : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.       : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
1000.    : Frequency in MHz
100.     : Antenna height in m
1        : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0        : Polarization (0=HOR, 1=VER)
0.       : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)

```

```

0.      : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0       : Number of cut-back angles and factors (used for specific height-finder antenna)
0       : Minimum output height in m
1000.   : Maximum output height in m
50.     : Maximum output range in km
20      : Number of output height points
1       : Number of output range points
0       : Extrapolation flag
0       : Surface absolute humidity in g/m3
0       : Surface air temperature in degrees
0       : Gaseous absorption attenuation rate in dB/km
0       : Number of wind speeds/ranges specified
1       : Number of refractivity profiles
2       : Number of levels in refractivity profiles
0       : Range of first refractivity profiles in km
0.      350.   : Height & M-unit value of ref. profile 1, level 1
1000.   468.   : Height & M-unit value of ref. profile 1, level 2
1       : Number of ground composition types
0., 0, 0., 0. : Range(km), ground type (integer), permittivity, conductivity
0       : Number of terrain range/height points

```

8.14 HORZ.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.false.  : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.       : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
1000.    : Frequency in MHz
25.      : Antenna height in m
1        : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0        : Polarization (0=HOR, 1=VER)
0.       : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.       : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0       : Number of cut-back angles and factors (used for specific height-finder antenna)
0       : Minimum output height in m
2000.    : Maximum output height in m
50.      : Maximum output range in km
20       : Number of output height points
1        : Number of output range points
0        : Extrapolation flag
0.       : Surface absolute humidity in g/m3
0.       : Surface air temperature in degrees
0.       : Gaseous absorption attenuation rate in dB/km
0       : Number of wind speeds/ranges specified
1       : Number of refractivity profiles
2       : Number of levels in refractivity profiles
0       : Range of first refractivity profiles in km
0.      350.   : Height & M-unit value of ref. profile 1, level 1
1000.   468.   : Height & M-unit value of ref. profile 1, level 2
1       : Number of ground composition types
0., 0, 0., 0. : Range(km), ground type (integer), permittivity, conductivity
0       : Number of terrain range/height points

```

8.15 HTFIND.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.false.  : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.       : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
1000.    : Frequency in MHz
25.      : Antenna height in m
5        : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0        : Polarization (0=HOR, 1=VER)
2.       : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)

```

```

0.      : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0       : Number of cut-back angles and factors (used for specific height-finder antenna)
0       : Minimum output height in m
2000.   : Maximum output height in m
50.     : Maximum output range in km
20      : Number of output height points
1       : Number of output range points
0       : Extrapolation flag
0.      : Surface absolute humidity in g/m3
0.      : Surface air temperature in degrees
0.      : Gaseous absorption attenuation rate in dB/km
0       : Number of wind speeds/ranges specified
1       : Number of refractivity profiles
2       : Number of levels in refractivity profiles
0.      : Range of first refractivity profiles in km
0.      350.   : Height & M-unit value of ref. profile 1, level 1
1000.   468.   : Height & M-unit value of ref. profile 1, level 2
1       : Number of ground composition types
0., 0, 0., 0. : Range(km), ground type (integer), permittivity, conductivity
0       : Number of terrain range/height points

```

8.16 LOBW.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.false.  : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.       : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
1000.    : Frequency in MHz
25.      : Antenna height in m
2        : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0        : Polarization (0=HOR, 1=VER)
.5       : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.       : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0        : Number of cut-back angles and factors (used for specific height-finder antenna)
0        : Minimum output height in m
2000.    : Maximum output height in m
50.      : Maximum output range in km
20       : Number of output height points
1        : Number of output range points
0        : Extrapolation flag
0.       : Surface absolute humidity in g/m3
0.       : Surface air temperature in degrees
0.       : Gaseous absorption attenuation rate in dB/km
0        : Number of wind speeds/ranges specified
1        : Number of refractivity profiles
2        : Number of levels in refractivity profiles
0.       : Range of first refractivity profiles in km
0.      350.   : Height & M-unit value of ref. profile 1, level 1
1000.   468.   : Height & M-unit value of ref. profile 1, level 2
1        : Number of ground composition types
0., 0, 0., 0. : Range(km), ground type (integer), permittivity, conductivity
0        : Number of terrain range/height points

```

8.17 LOEL.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.false.  : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.       : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
1000.    : Frequency in MHz
25.      : Antenna height in m
2        : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0        : Polarization (0=HOR, 1=VER)
1.       : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
-10.    : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)

```

```

0      : Number of cut-back angles and factors (used for specific height-finder antenna)
0.     : Minimum output height in m
20000. : Maximum output height in m
50.    : Maximum output range in km
20     : Number of output height points
1      : Number of output range points
0      : Extrapolation flag
0.     : Surface absolute humidity in g/m3
0.     : Surface air temperature in degrees
0.     : Gaseous absorption attenuation rate in dB/km
0      : Number of wind speeds/ranges specified
1      : Number of refractivity profiles
2      : Number of levels in refractivity profiles
0.     : Range of first refractivity profiles in km
0.     350. : Height & M-unit value of ref. profile 1, level 1
1000.  468. : Height & M-unit value of ref. profile 1, level 2
1      : Number of ground composition types
0., 0, 0., 0. : Range(km), ground type (integer), permittivity, conductivity
0      : Number of terrain range/height points

```

8.18 LOFREQ.IN

```

.true. : LERR6 error flag
.true. : LERR12 error flag
.false. : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.     : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.     : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false. : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
100.   : Frequency in MHz
25.    : Antenna height in m
1      : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0      : Polarization (0=HOR, 1=VER)
0.     : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.     : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0      : Number of cut-back angles and factors (used for specific height-finder antenna)
0.     : Minimum output height in m
5000.  : Maximum output height in m
50.    : Maximum output range in km
20     : Number of output height points
1      : Number of output range points
0      : Extrapolation flag
0.     : Surface absolute humidity in g/m3
0.     : Surface air temperature in degrees
0.     : Gaseous absorption attenuation rate in dB/km
0      : Number of wind speeds/ranges specified
1      : Number of refractivity profiles
2      : Number of levels in refractivity profiles
0.     : Range of first refractivity profiles in km
0.     350. : Height & M-unit value of ref. profile 1, level 1
1000.  468. : Height & M-unit value of ref. profile 1, level 2
1      : Number of ground composition types
0., 0, 0., 0. : Range(km), ground type (integer), permittivity, conductivity
0      : Number of terrain range/height points

```

8.19 LOTRAN.IN

```

.true. : LERR6 error flag
.true. : LERR12 error flag
.false. : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.     : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.     : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false. : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
1000.  : Frequency in MHz
1.5    : Antenna height in m
1      : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0      : Polarization (0=HOR, 1=VER)
0.     : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.     : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0      : Number of cut-back angles and factors (used for specific height-finder antenna)

```

```

0.      : Minimum output height in m
10000.  : Maximum output height in m
50.     : Maximum output range in km
20      : Number of output height points
1       : Number of output range points
0       : Extrapolation flag
0.      : Surface absolute humidity in g/m3
0.      : Surface air temperature in degrees
0.      : Gaseous absorption attenuation rate in dB/km
0       : Number of wind speeds/ranges specified
1       : Number of refractivity profiles
2       : Number of levels in refractivity profiles
0.      : Range of first refractivity profiles in km
0.      350.    : Height & M-unit value of ref. profile 1, level 1
1000.   468.    : Height & M-unit value of ref. profile 1, level 2
1       : Number of ground composition types
0., 0, 0., 0.  : Range(km), ground type (integer), permittivity, conductivity
0       : Number of terrain range/height points

```

8.20 MPRT.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.false.  : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.       : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
300.     : Frequency in MHz
800.     : Antenna height in m
2        : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
1        : Polarization (0=HOR, 1=VER)
0.5      : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
-2.5     : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0        : Number of cut-back angles and factors (used for specific height-finder antenna)
0.       : Minimum output height in m
1100.    : Maximum output height in m
60.      : Maximum output range in km
1        : Number of output height points
30       : Number of output range points
0        : Extrapolation flag
7.5      : Surface absolute humidity in g/m3
0.       : Surface air temperature in degrees
0.       : Gaseous absorption attenuation rate in dB/km
0        : Number of wind speeds/ranges specified
1        : Number of refractivity profiles
2        : Number of levels in refractivity profiles
0.       : Range of first refractivity profiles in km
0.      350.    : Height & M-unit value of ref. profile 1, level 1
1000.   468.    : Height & M-unit value of ref. profile 1, level 2
1        : Number of ground composition types
0., 7, 7.5, 0.01 : Range(km), ground type (integer), permittivity, conductivity
5        : Number of terrain range(km)/height points
0.       0
10.0     0
30.0     600
50.0     0
60.0     0

```

8.21 PERW.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.true.   : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
10.      : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
300.     : Frequency in MHz
10.      : Antenna height in m
1        : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
1        : Polarization (0=HOR, 1=VER)

```

```

180.      : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.        : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0         : Number of cut-back angles and factors (used for specific height-finder antenna)
0         : Minimum output height in m
1000.     : Maximum output height in m
50.       : Maximum output range in km
1         : Number of output height points
20        : Number of output range points
0         : Extrapolation flag
7.5       : Surface absolute humidity in g/m3
0.        : Surface air temperature in degrees
0.        : Gaseous absorption attenuation rate in dB/km
0         : Number of wind speeds/ranges specified
1         : Number of refractivity profiles
2         : Number of levels in refractivity profiles
0.        : Range of first refractivity profiles in km
0.        350.      : Height & M-unit value of ref. profile 1, level 1
1000.    468.      : Height & M-unit value of ref. profile 1, level 2
1         : Number of ground composition types
0., 7, 7.5, 0.01   : Range(km), ground type (integer), permittivity, conductivity
11        : Number of terrain range(km)/height points
0.        0
18.750    0
20.312    210
21.875    320
23.4375   375
25.000    390
26.5625   375
28.125    320
31.250    90
32.8125   0
50.000    0

```

8.22 PVT.IN

```

.true.    : LERR6 error flag
.true.    : LERR12 error flag
.false.   : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.        : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.        : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.   : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
500.      : Frequency in MHz
10.       : Antenna height in m
1         : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
1         : Polarization (0=HOR, 1=VER)
180.      : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.        : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0         : Number of cut-back angles and factors (used for specific height-finder antenna)
0.        : Minimum output height in m
2000.     : Maximum output height in m
10.       : Maximum output range in km
20        : Number of output height points
1         : Number of output range points
0         : Extrapolation flag
7.5       : Surface absolute humidity in g/m3
0.        : Surface air temperature in degrees
0.        : Gaseous absorption attenuation rate in dB/km
0         : Number of wind speeds/ranges specified
1         : Number of refractivity profiles
2         : Number of levels in refractivity profiles
0.        : Range of first refractivity profiles in km
0.        350.      : Height & M-unit value of ref. profile 1, level 1
1000.    468.      : Height & M-unit value of ref. profile 1, level 2
1         : Number of ground composition types
0., 7, 7.5, 0.01   : Range(km), ground type (integer), permittivity, conductivity
17        : Number of terrain range(km)/height points
0.        625
3.170     476
6.340     347
9.510     239
12.690    151
15.870    83
19.040    35

```


22.220 7
 25.000 0
 27.780 7
 30.960 35
 34.130 83
 37.310 151
 40.490 239
 43.660 347
 46.830 476
 50.000 625

8.23 RDLONGB.IN

```
.true.      : LERR6 error flag
.true.      : LERR12 error flag
.false.     : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.          : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.          : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.     : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
150.        : Frequency in MHz
100.        : Antenna height in m
1           : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0           : Polarization (0=HOR, 1=VER)
1.          : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.          : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0           : Number of cut-back angles and factors (used for specific height-finder antenna)
0.          : Minimum output height in m
1000.       : Maximum output height in m
100.        : Maximum output range in km
20          : Number of output height points
1           : Number of output range points
0           : Extrapolation flag
0.          : Surface absolute humidity in g/m3
0.          : Surface air temperature in degrees
0.          : Gaseous absorption attenuation rate in dB/km
0           : Number of wind speeds/ranges specified
2           : Number of refractivity profiles
4           : Number of levels in refractivity profiles
0.          : Range of first refractivity profiles in km
0.          350.    : Height & M-unit value of ref. profile 1, level 1
0.          350.    : Height & M-unit value of ref. profile 1, level 2
0.          350.    : Height & M-unit value of ref. profile 1, level 3
1000.       468.    : Height & M-unit value of ref. profile 1, level 4
100.        : Range of second refractivity profiles in km
0.          339.    : Height & M-unit value of ref. profile 2, level 1
250.        368.5   : Height & M-unit value of ref. profile 2, level 2
300.        319.    : Height & M-unit value of ref. profile 2, level 3
1000.       401.6   : Height & M-unit value of ref. profile 2, level 4
6           : Number of ground composition types
0., 2, 0., 0.      : Range(km), ground type (integer), permittivity, conductivity
28.500, 0, 0., 0.  : Range(km), ground type (integer), permittivity, conductivity
64.800, 3, 0., 0.  : Range(km), ground type (integer), permittivity, conductivity
68.700, 0, 0., 0.  : Range(km), ground type (integer), permittivity, conductivity
74.100, 4, 0., 0.  : Range(km), ground type (integer), permittivity, conductivity
100.200, 0, 0., 0. : Range(km), ground type (integer), permittivity, conductivity
167         : Number of terrain range/height points
0.000       8      : Range & height of terrain point 1 in km
0.300       8
0.600       9
0.900       9
1.200      10
1.500      11
1.800      12
2.100      13
2.400      14
2.700      15      : Range & height of terrain point 10 in km
3.000      17
3.300      19
3.600      21
3.900      23
4.200      25
4.500      27
4.800      28
```

5.100	30	
5.400	31	
5.700	31	: Range & height of terrain point 20 in km
6.000	29	
6.300	23	
6.600	14	
6.900	9	
7.200	7	
7.500	7	
7.800	9	
8.100	11	
8.400	14	
8.700	13	: Range & height of terrain point 30 in km
9.300	13	
9.600	12	
9.900	11	
10.200	8	
10.800	8	
11.100	7	
12.600	7	
12.900	6	
14.400	6	
14.700	7	: Range & height of terrain point 40 in km
15.000	8	
15.300	8	
15.600	9	
15.900	10	
16.200	11	
16.500	11	
16.800	12	
17.400	12	
17.700	13	
18.000	13	: Range & height of terrain point 50 in km
18.300	14	
18.600	15	
18.900	16	
19.200	18	
19.500	20	
19.800	21	
20.100	22	
20.400	23	
20.700	24	
21.000	24	: Range & height of terrain point 60 in km
21.300	25	
21.600	26	
21.900	27	
22.200	27	
22.500	28	
22.800	29	
23.400	29	
23.700	30	
24.600	30	
24.900	32	: Range & height of terrain point 70 in km
25.200	34	
25.500	38	
26.100	38	
26.400	36	
26.700	34	
27.000	32	
27.300	27	
27.600	15	
27.900	6	
28.200	1	: Range & height of terrain point 80 in km
28.500	0	
64.500	0	
64.800	8	
65.100	30	
65.400	39	
65.700	61	
66.600	61	
66.900	24	
67.200	14	
67.500	26	: Range & height of terrain point 90 in km
67.800	16	
68.100	1	
68.400	1	

68.700	0	
73.800	0	
74.100	1	
74.400	1	
74.700	10	
75.000	8	
75.300	39	: Range & height of terrain point 100 in km
75.600	45	
75.900	53	
76.200	61	
76.500	61	
76.800	82	
77.100	61	
77.400	78	
77.700	61	
78.000	129	
78.300	30	: Range & height of terrain point 110 in km
78.600	46	
78.900	159	
79.200	184	
79.500	226	
79.800	152	
80.100	201	
80.400	244	
80.700	152	
81.000	143	
81.300	91	: Range & height of terrain point 120 in km
81.600	107	
81.900	152	
82.200	152	
82.500	170	
82.800	152	
83.100	66	
83.400	70	
83.700	121	
84.000	152	
84.300	170	: Range & height of terrain point 130 in km
84.600	141	
84.900	139	
85.200	147	
85.500	177	
85.800	152	
86.100	61	
86.700	61	
87.000	70	
87.300	44	
87.600	11	: Range & height of terrain point 140 in km
87.900	1	
89.400	1	
89.700	61	
90.000	84	
90.300	152	
90.600	152	
90.900	101	
91.200	40	
91.500	15	
91.800	20	: Range & height of terrain point 150 in km
92.100	2	
92.400	10	
92.700	4	
93.000	1	
93.300	1	
93.600	0	
93.900	1	
96.300	1	
96.600	0	
96.900	1	: Range & height of terrain point 160 in km
97.500	1	
97.800	2	
98.100	3	
99.300	3	
99.600	2	
99.900	2	
100.200	1	: Range & height of terrain point 167 in km

8.24 RNGDEP.IN

```
.true.      : LERR6 error flag
.true.      : LERR12 error flag
.false.     : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.          : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.          : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.     : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
3000.       : Frequency in MHz
25.         : Antenna height in m
1           : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0           : Polarization (0=HOR, 1=VER)
5.          : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.          : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0           : Number of cut-back angles and factors (used for specific height-finder antenna)
0.          : Minimum output height in m
2000.       : Maximum output height in m
250.        : Maximum output range in km
20          : Number of output height points
1           : Number of output range points
0           : Extrapolation flag
0.          : Surface absolute humidity in g/m3
0.          : Surface air temperature in degrees
0.          : Gaseous absorption attenuation rate in dB/km
0           : Number of wind speeds/ranges specified
2           : Number of refractivity profiles
4           : Number of levels in refractivity profiles
0.          : Range of first refractivity profiles in km
0.    330.   : Height & M-unit value of ref. profile 1, level 1
100.  342.5  : Height & M-unit value of ref. profile 1, level 2
230.  312.5  : Height & M-unit value of ref. profile 1, level 3
2000. 517.8  : Height & M-unit value of ref. profile 1, level 4
250.        : Range of second refractivity profiles in km
0.    330.   : Height & M-unit value of ref. profile 2, level 1
600.  405.   : Height & M-unit value of ref. profile 2, level 2
730.  375.   : Height & M-unit value of ref. profile 2, level 3
2000. 522.3  : Height & M-unit value of ref. profile 2, level 4
1           : Number of ground composition types
0., 0, 0., 0. : Range(km), ground type (integer), permittivity, conductivity
0           : Number of terrain range(km)/height points
```

8.25 SBDUCT.IN

```
.true.      : LERR6 error flag
.true.      : LERR12 error flag
.false.     : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.          : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.          : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.     : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
3000.       : Frequency in MHz
25.         : Antenna height in m
2           : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0           : Polarization (0=HOR, 1=VER)
5.          : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.          : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0           : Number of cut-back angles and factors (used for specific height-finder antenna)
0.          : Minimum output height in m
5000.       : Maximum output height in m
200.        : Maximum output range in m
20          : Number of output height points
1           : Number of output range points
0           : Extrapolation flag
0.          : Surface absolute humidity in g/m3
0.          : Surface air temperature in degrees
0.          : Gaseous absorption attenuation rate in dB/km
0           : Number of wind speeds/ranges specified
1           : Number of refractivity profiles
4           : Number of levels in refractivity profiles
0.          : Range of first refractivity profiles in km
0.    339.0   : Height & M-unit value of ref. profile 1, level 1
```

```

250.    368.5      : Height & M-unit value of ref. profile 1, level 2
300.    319.0      : Height & M-unit value of ref. profile 1, level 3
1000.   401.6      : Height & M-unit value of ref. profile 1, level 4
1       : Number of ground composition types
0., 0, 0., 0.    : Range(km), ground type (integer), permittivity, conductivity
0       : Number of terrain range/height points

```

8.26 SBDUCTRF.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.false.  : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.       : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
3000.    : Frequency in MHz
25.      : Antenna height in m
2        : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
1        : Polarization (0=HOR, 1=VER)
5.       : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.       : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0        : Number of cut-back angles and factors (used for specific height-finder antenna)
0.       : Minimum output height in m
1000.    : Maximum output height in m
200.     : Maximum output range in km
20       : Number of output height points
1        : Number of output range points
0        : Extrapolation flag
0.       : Surface absolute humidity in g/m3
0.       : Surface air temperature in degrees
0.       : Gaseous absorption attenuation rate in dB/km
1        : Number of wind speeds/ranges specified
10., 0.  : Wind speed (m/s), Range(km)
1        : Number of refractivity profiles
4        : Number of levels in refractivity profiles
0.       : Range of first refractivity profiles in km
0.       339.0     : Height & M-unit value of ref. profile 1, level 1
250.     368.5     : Height & M-unit value of ref. profile 1, level 2
300.     319.0     : Height & M-unit value of ref. profile 1, level 3
1000.    401.6     : Height & M-unit value of ref. profile 1, level 4
1        : Number of ground composition types
0., 0, 0., 0.    : Range(km), ground type (integer), permittivity, conductivity
0        : Number of terrain range/height points

```

8.27 SINEX.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.false.  : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.       : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
1000.    : Frequency in MHz
25.      : Antenna height in m
3        : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0        : Polarization (0=HOR, 1=VER)
1.       : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.       : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0        : Number of cut-back angles and factors (used for specific height-finder antenna)
0.       : Minimum output height in m
2000.    : Maximum output height in m
50.0     : Maximum output range in km
20       : Number of output height points
1        : Number of output range points
0        : Extrapolation flag
0.       : Surface absolute humidity in g/m3
0.       : Surface air temperature in degrees
0.       : Gaseous absorption attenuation rate in dB/km
0        : Number of wind speeds/ranges specified
1        : Number of refractivity profiles

```

```

2      : Number of levels in refractivity profiles
0.     : Range of first refractivity profiles in km
0.     350.      : Height & M-unit value of ref. profile 1, level 1
1000.  468.     : Height & M-unit value of ref. profile 1, level 2
1      : Number of ground composition types
0., 0, 0., 0.   : Range(km), ground type (integer), permittivity, conductivity
0      : Number of terrain range/height points

```

8.28 TROPOS.IN

```

.true.      : LERR6 error flag
.true.      : LERR12 error flag
.false.     : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.          : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.          : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.true.      : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
100.        : Frequency in MHz
25.         : Antenna height in m
1           : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0           : Polarization (0=HOR, 1=VER)
0.          : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.          : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0           : Number of cut-back angles and factors (used for specific height-finder antenna)
0.          : Minimum output height in m
2000.       : Maximum output height in m
200.        : Maximum output range in m
20          : Number of output height points
1           : Number of output range points
0           : Extrapolation flag
0.          : Surface absolute humidity in g/m3
0.          : Surface air temperature in degrees
0.          : Gaseous absorption attenuation rate in dB/km
0           : Number of wind speeds/ranges specified
1           : Number of refractivity profiles
2           : Number of levels in refractivity profiles
0.          : Range of first refractivity profiles in km
0.          350.      : Height & M-unit value of ref. profile 1, level 1
1000.       468.     : Height & M-unit value of ref. profile 1, level 2
1           : Number of ground composition types
0., 0, 0., 0.   : Range(km), ground type (integer), permittivity, conductivity
0           : Number of terrain range/height points

```

8.29 TROPOT.IN

```

.true.      : LERR6 error flag
.true.      : LERR12 error flag
.false.     : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.          : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.          : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.true.      : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
100.        : Frequency in MHz
25.         : Antenna height in m
1           : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0           : Polarization (0=HOR, 1=VER)
0.          : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.          : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0           : Number of cut-back angles and factors (used for specific height-finder antenna)
0.          : Minimum output height in m
2000.       : Maximum output height in m
200.        : Maximum output range in m
20          : Number of output height points
1           : Number of output range points
0           : Extrapolation flag
0.          : Surface absolute humidity in g/m3
0.          : Surface air temperature in degrees
0.          : Gaseous absorption attenuation rate in dB/km
0           : Number of wind speeds/ranges specified
1           : Number of refractivity profiles
2           : Number of levels in refractivity profiles

```

```

0.      : Range of first refractivity profiles in km
0.      350.      : Height & M-unit value of ref. profile 1, level 1
1000.   468.      : Height & M-unit value of ref. profile 1, level 2
6       : Number of ground composition types
0., 2, 0., 0.    : Range(km), ground type (integer), permittivity, conductivity
28.500, 0, 0., 0.
64.800, 3, 0., 0.
68.700, 0, 0., 0.
74.100, 4, 0., 0.
100.200, 0, 0., 0.
169     : Number of terrain range/height points
0.000    8      : Range(km) & height(m) of terrain point 1
0.300    8
0.600    9
0.900    9
1.200   10
1.500   11
1.800   12
2.100   13
2.400   14
2.700   15      : Range(km) & height(m) of terrain point 10
3.000   17
3.300   19
3.600   21
3.900   23
4.200   25
4.500   27
4.800   28
5.100   30
5.400   31
5.700   31      : Range(km) & height(m) of terrain point 20
6.000   29
6.300   23
6.600   14
6.900    9
7.200    7
7.500    7
7.800    9
8.100   11
8.400   14
8.700   13      : Range(km) & height(m) of terrain point 30
9.300   13
9.600   12
9.900   11
10.200    8
10.800    8
11.100    7
12.600    7
12.900    6
14.400    6
14.700    7      : Range(km) & height(m) of terrain point 40
15.000    8
15.300    8
15.600    9
15.900   10
16.200   11
16.500   11
16.800   12
17.400   12
17.700   13
18.000   13      : Range(km) & height(m) of terrain point 50
18.300   14
18.600   15
18.900   16
19.200   18
19.500   20
19.800   21
20.100   22
20.400   23
20.700   24
21.000   24      : Range(km) & height(m) of terrain point 60
21.300   25
21.600   26
21.900   27
22.200   27
22.500   28

```

22.800	29	
23.400	29	
23.700	30	
24.600	30	
24.900	32	: Range(km) & height(m) of terrain point 70
25.200	34	
25.500	38	
26.100	38	
26.400	36	
26.700	34	
27.000	32	
27.300	27	
27.600	15	
27.900	6	
28.200	1	: Range(km) & height(m) of terrain point 80
28.500	0	
64.500	0	
64.800	8	
65.100	30	
65.400	39	
65.700	61	
66.600	61	
66.900	24	
67.200	14	
67.500	26	: Range(km) & height(m) of terrain point 90
67.800	16	
68.100	1	
68.400	1	
68.700	0	
73.800	0	
74.100	1	
74.400	1	
74.700	10	
75.000	8	
75.300	39	: Range(km) & height(m) of terrain point 100
75.600	45	
75.900	53	
76.200	61	
76.500	61	
76.800	82	
77.100	61	
77.400	78	
77.700	61	
78.000	129	
78.300	30	: Range(km) & height(m) of terrain point 110
78.600	46	
78.900	159	
79.200	184	
79.500	226	
79.800	152	
80.100	201	
80.400	244	
80.700	152	
81.000	143	
81.300	91	: Range(km) & height(m) of terrain point 120
81.600	107	
81.900	152	
82.200	152	
82.500	170	
82.800	152	
83.100	66	
83.400	70	
83.700	121	
84.000	152	
84.300	170	: Range(km) & height(m) of terrain point 130
84.600	141	
84.900	139	
85.200	147	
85.500	177	
85.800	152	
86.100	61	
86.700	61	
87.000	70	
87.300	44	
87.600	11	: Range(km) & height(m) of terrain point 140
87.900	1	


```

89.400      1
89.700      61
90.000      84
90.300     152
90.600     152
90.900     101
91.200      40
91.500      15
91.800      20      : Range(km) & height(m) of terrain point 150
92.100       2
92.400      10
92.700       4
93.000       1
93.300       1
93.600       0
93.900       1
96.300       1
96.600       0
96.900       1      : Range(km) & height(m) of terrain point 160
97.500       1
97.800       2
98.100       3
99.300       3
99.600       2
99.900       2
100.200      1
100.200      0.
200.000      0.      : Range(km) & height(m) of terrain point 167

```

8.30 USERDEFA

```

.true.      : LERR6 error flag
.true.      : LERR12 error flag
.false.     : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.          : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.          : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.     : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
900.        : Frequency in MHz
6.          : Antenna height in m
7.          : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0.          : Polarization (0=HOR, 1=VER)
0.          : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
2.          : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
54          : Number of angle/factor pairs (used for antenna types 6 and 7)
-17         0.017
-16         0.044
-15         0.080
-14         0.126
-13         0.182
-12         0.245
-11         0.316
-10         0.389
-9          0.479
-8          0.556
-7          0.631
-6          0.716
-5          0.785
-4          0.861
-3          0.912
-2          0.966
-1          0.998
0           1.000
1           1.000
2           0.966
3           0.902
4           0.822
5           0.742
6           0.646
7           0.569
8           0.501
9           0.452
10          0.422
11          0.402

```

```

12      0.389
13      0.375
14      0.359
15      0.339
16      0.305
17      0.276
18      0.245
19      0.221
20      0.210
21      0.199
22      0.190
23      0.180
24      0.164
25      0.148
26      0.130
27      0.110
28      0.095
29      0.077
30      0.070
31      0.065
32      0.058
33      0.050
34      0.039
35      0.031
36      0.025
0.      : Minimum output height in m
3000.   : Maximum output height in m
300.    : Maximum output range in km
20      : Number of output height points
1       : Number of output range points
0       : Extrapolation flag
0.      : Surface absolute humidity in g/m3
0.      : Surface air temperature in degrees
0.      : Gaseous absorption attenuation rate in dB/km
0       : Number of wind speeds/ranges specified
1       : Number of refractivity profiles
4       : Number of levels in refractivity profiles
0.      : Range of first refractivity profiles in km
0.      339.0      : Height & M-unit value of ref. profile 1, level 1
250.    368.5      : Height & M-unit value of ref. profile 1, level 2
300.    319.0      : Height & M-unit value of ref. profile 1, level 3
1000.   401.6      : Height & M-unit value of ref. profile 1, level 4
1       : Number of ground composition types
0., 0, 0., 0.    : Range(km), ground type (integer), permittivity, conductivity
0       : Number of terrain range/height points

```

8.31 USERHF.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.false.  : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.       : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
1000.    : Frequency in MHz
25.      : Antenna height in m
6        : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0        : Polarization (0=HOR, 1=VER)
1.       : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.       : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
10       : Number of angle/factor pairs (used for antenna types 6 and 7)
1.0 0.9
1.5 0.8
2.0 0.7
2.5 0.6
3.0 0.5
3.5 0.4
4.0 0.3
4.5 0.2
5.0 0.1
5.5 0.0
0.       : Minimum output height in m
2000.    : Maximum output height in m

```

```

50.0      : Maximum output range in km
20        : Number of output height points
1         : Number of output range points
0         : Extrapolation flag
0         : Surface absolute humidity in g/m3
0         : Surface air temperature in degrees
0         : Gaseous absorption attenuation rate in dB/km
0         : Number of wind speeds/ranges specified
1         : Number of refractivity profiles
2         : Number of levels in refractivity profiles
0         : Range of first refractivity profiles in km
0         350.      : Height & M-unit value of ref. profile 1, level 1
1000.    468.      : Height & M-unit value of ref. profile 1, level 2
1         : Number of ground composition types
0., 0, 0., 0.    : Range(km), ground type (integer), permittivity, conductivity
0         : Number of terrain range/height points

```

8.32 VERT.IN

```

.true.    : LERR6 error flag
.true.    : LERR12 error flag
.false.   : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0         : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1         : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.   : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
1000.     : Frequency in MHz
25.       : Antenna height in m
1         : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
1         : Polarization (0=HOR, 1=VER)
0         : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0         : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0         : Number of angle/factor pairs (used for antenna types 6 and 7)
0         : Minimum output height in m
2000.     : Maximum output height in m
50.       : Maximum output range in km
20        : Number of output height points
1         : Number of output range points
0         : Extrapolation flag
0         : Surface absolute humidity in g/m3
0         : Surface air temperature in degrees
0         : Gaseous absorption attenuation rate in dB/km
0         : Number of wind speeds/ranges specified
1         : Number of refractivity profiles
2         : Number of levels in refractivity profiles
0         : Range of first refractivity profiles in km
0         350.      : Height & M-unit value of ref. profile 1, level 1
1000.    468.      : Height & M-unit value of ref. profile 1, level 2
1         : Number of ground composition types
0., 0, 0., 0.    : Range(km), ground type (integer), permittivity, conductivity
0         : Number of terrain range/height points

```

8.33 VERTMIX.IN

```

.true.    : LERR6 error flag
.true.    : LERR12 error flag
.false.   : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0         : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1         : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.   : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
100.      : Frequency in MHz
10.       : Antenna height in m
1         : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
1         : Polarization (0=HOR, 1=VER)
1         : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0         : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0         : Number of angle/factor pairs (used for antenna types 6 and 7)
0         : Minimum output height in m
1000.     : Maximum output height in m
50.       : Maximum output range in km
20        : Number of output height points

```

```

1      : Number of output range points
0      : Extrapolation flag
0.     : Surface absolute humidity in g/m3
0.     : Surface air temperature in degrees
0.     : Gaseous absorption attenuation rate in dB/km
0      : Number of wind speeds/ranges specified
1      : Number of refractivity profiles
2      : Number of levels in refractivity profiles
0.     : Range of first refractivity profiles in km
0.     350.      : Height & M-unit value of ref. profile 1, level 1
1000.  468.     : Height & M-unit value of ref. profile 1, level 2
2      : Number of ground composition types
0., 4, 0., 0.   : Range(km), ground type (integer), permittivity, conductivity
25.0, 0, 0., 0. : Range(km), ground type (integer), permittivity, conductivity
2      : Number of terrain range/height points
0.     0.       : Range(km) & height(m) of terrain point 1
50.    0.       : Range(km) & height(m) of terrain point 2

```

8.34 VERTSEA.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.false.  : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.       : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
100.     : Frequency in MHz
25.      : Antenna height in m
1        : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
1        : Polarization (0=HOR, 1=VER)
1.       : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.       : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0        : Number of cut-back angles and factors (used for specific height-finder antenna)
0.       : Minimum output height in m
1000.    : Maximum output height in m
300.     : Maximum output range in km
20       : Number of output height points
1        : Number of output range points
0        : Extrapolation flag
0.       : Surface absolute humidity in g/m3
0.       : Surface air temperature in degrees
0.       : Gaseous absorption attenuation rate in dB/km
0        : Number of wind speeds/ranges specified
1        : Number of refractivity profiles
4        : Number of levels in refractivity profiles
0.       : Range of first refractivity profiles in km
0.     339.0    : Height & M-unit value of ref. profile 1, level 1
250.    368.5   : Height & M-unit value of ref. profile 1, level 2
300.    319.0   : Height & M-unit value of ref. profile 1, level 3
1000.   401.6   : Height & M-unit value of ref. profile 1, level 4
1        : Number of ground composition types
0., 0, 0., 0.  : Range(km), ground type (integer), permittivity, conductivity
0        : Number of terrain range/height points

```

8.35 VERTUSRD.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.false.  : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.       : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
100.     : Frequency in MHz
10.      : Antenna height in m
1        : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
1        : Polarization (0=HOR, 1=VER)
1.       : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.       : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0        : Number of cut-back angles and factors (used for specific height-finder antenna)

```

```

0.      : Minimum output height in m
1000.   : Maximum output height in m
50.     : Maximum output range in km
20      : Number of output height points
1       : Number of output range points
0       : Extrapolation flag
0.      : Surface absolute humidity in g/m3
0.      : Surface air temperature in degrees
0.      : Gaseous absorption attenuation rate in dB/km
0       : Number of wind speeds/ranges specified
1       : Number of refractivity profiles
2       : Number of levels in refractivity profiles
0.      : Range of first refractivity profiles in km
0.      350.   : Height & M-unit value of ref. profile 1, level 1
1000.   468.   : Height & M-unit value of ref. profile 1, level 2
1       : Number of ground composition types
0., 7, 3., 6.e-4 : Range(km), ground type (integer), permittivity, conductivity
2       : Number of terrain range/height points
0.      0.     : Range(km) & height(m) of terrain point 1
50.     0.     : Range(km) & height(m) of terrain point 2

```

8.36 WEDGE.IN

```

.true.   : LERR6 error flag
.true.   : LERR12 error flag
.false.  : Perform PE calcs only? ('.true.' = yes, '.false.' = no)
0.       : Maximum PE calculation angle in degrees (required if using PE calcs only, ignored otherwise)
1.       : PE range step multiplier (required if using PE calcs only, ignored otherwise)
.false.  : Troposcatter flag: '.false.'=no troposcatter, '.true.'=troposcatter
1000.    : Frequency in MHz
25.      : Antenna height in m
1        : Antenna type (1=OMNI,2=GAUSS,3=SINC(X),4=COSEC2,5=HTFIND,6=USRHTFIND,7=USRDEF)
0        : Polarization (0=HOR, 1=VER)
1.       : Beamwidth in deg (this value is ignored for OMNI and USRDEF antenna)
0.       : Antenna elevation angle in deg (this value is ignored for OMNI and USRDEF antenna)
0        : Number of cut-back angles and factors (used for specific height-finder antenna)
0.       : Minimum output height in m
1000.    : Maximum output height in m
100.     : Maximum output range in km
20       : Number of output height points
1        : Number of output range points
0        : Extrapolation flag
0.       : Surface absolute humidity in g/m3
0.       : Surface air temperature in degrees
0.       : Gaseous absorption attenuation rate in dB/km
0        : Number of wind speeds/ranges specified
1        : Number of refractivity profiles
2        : Number of levels in refractivity profiles
0.       : Range of first refractivity profiles in km
0.      350.   : Height & M-unit value of ref. profile 1, level 1
1000.   468.   : Height & M-unit value of ref. profile 1, level 2
1        : Number of ground composition types
0., 0, 0., 0.  : Range(km), ground type (integer), permittivity, conductivity
5        : Number of terrain range/height points
0.      0.     : Range(km) & height(m) of terrain point 1
45.0    0.     : Range(km) & height(m) of terrain point 2
50.0    200.   : Range(km) & height(m) of terrain point 3
55.0    0.     : Range(km) & height(m) of terrain point 4
100.0   0.     : Range(km) & height(m) of terrain point 5

```

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-01-0188	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden to Department of Defense, Washington Headquarters Services Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 08-2002		2. REPORT TYPE Final		3. DATES COVERED (From - To)	
4. TITLE AND SUBTITLE ADVANCED PROPAGATION MODEL (APM) VER. 1.3.1 COMPUTER SOFTWARE CONFIGURATION ITEM (CSCI) DOCUMENTS				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER 0603207N	
				5d. PROJECT NUMBER	
6. AUTHORS A. E. Barrios W. L. Patterson				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) SSC San Diego San Diego, CA 92152-5001				8. PERFORMING ORGANIZATION REPORT NUMBER TD 3145	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Space and Naval Warfare Systems Command PMW-155 4301 Pacific Highway San Diego, CA 92110-3127				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.					
13. SUPPLEMENTARY NOTES This is the work of the United States Government and therefore is not copyrighted. This work may be copied and disseminated without restriction. Many SSC San Diego public release documents are available in electronic format at: http://www.spawar.navy.mil/sti/publications/pubs/index.html					
14. ABSTRACT This document describes the Advanced Propagation Model (APM) Version 1.3.1 computer software configuration item (CSCI) design and provides an input software requirement overview, a CSCI design architecture overview, and a detailed design description of each CSCI component.					
15. SUBJECT TERMS Mission Area: Command, Control, and Communications electromagnetic propagation software requirements tactical fleet applications					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			A. E. Barrios
U	U	U	UU	494	19b. TELEPHONE NUMBER (Include area code) (619) 553-1429

INITIAL DISTRIBUTION

20012	Patent Counsel	(1)
20271	Archive/Stock	(1)
20274	Library	(1)
2027	M. E. Cathcart	(1)
20271	E. R. Ratliff	(1)
20271	D. Richter	(1)
285	C. Sayre	(1)
2858	A. Barrios	(1)
2858	W. Patterson	(1)
2858	R. Paulus	(1)

Defense Technical Information Center
Fort Belvoir, VA 22060-6218

SSC San Diego Liaison Office
C/O PEO-SCS
Arlington, VA 22202-4804

Center for Naval Analyses
Alexandria, VA 22311-1850

Office of Naval Research
ATTN: NARDIC (Code 362)
Arlington, VA 22217-5660

Government-Industry Data Exchange
Program Operations Center
Corona, CA 91718-8000

Approved for public release; distribution is unlimited.