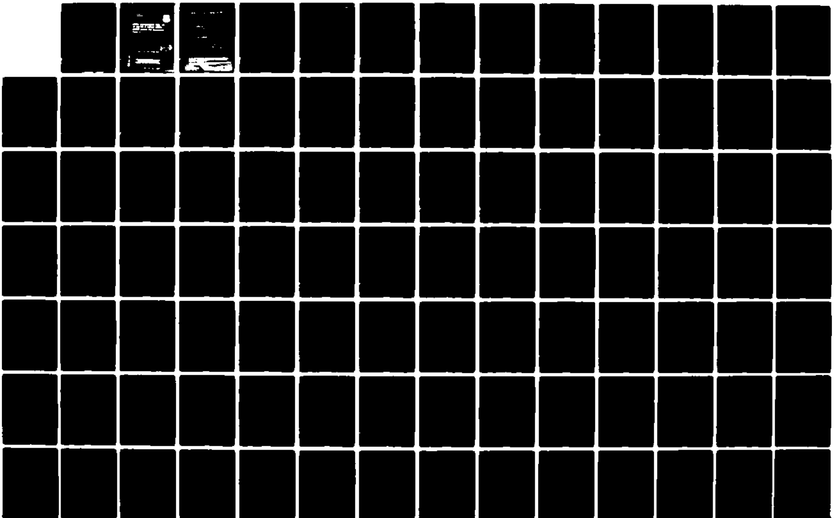
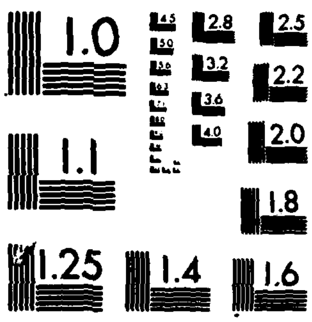
AD-A137 463 GENERAL ELECTROMAGNETIC MODEL FOR THE ANALYSIS OF COMPLEX SYSTEMS (GEMACS..(U) BDM CORP ALBUQUERQUE NM D L KADLEC ET AL. SEP 83 BDM/A-83-020-TR 1/5

UNCLASSIFIED RADC-TR-83-217-VOL-3 F30602-81-C-0084 F/G 20/14 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: RADC-TR-83-217, Vol III, Part 1	2. GOVT ACCESSION NO.: AD-A137 463
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): GENERAL ELECTROMAGNETIC MODEL FOR THE ANALYSIS OF COMPLEX SYSTEMS (GEMACS)	5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report February 81 - July 83
	6. PERFORMING ORG. REPORT NUMBER: BDM/A-83-020-TR
7. AUTHOR(s): Dr. Diana L. Kadlec, Dr. Edgar L. Coffey	8. CONTRACT OR GRANT NUMBER(s): F30602-81-C-0084
9. PERFORMING ORGANIZATION NAME AND ADDRESS: The BDM Corporation, 1801 Randolph Road, S.E., Albuquerque NM 87106	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 62702F, 23380333
11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (RBCT), Griffiss AFB NY 13441	12. REPORT DATE: September 1983
	13. NUMBER OF PAGES: 427
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Kenneth R. Siarkiewicz (RBCT)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Electromagnetic Compatibility
Method of Moments (MOM)
Geometrical Theory of Diffraction (GTD)
Antenna Analysis
Matrix Equation Solution
MOM/GTD Hybridization
EM Radiation and Scattering

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

GEMACS solves electromagnetic radiation and scattering problems. The Method of Moments (MOM) and Geometrical Theory of Diffraction (GTD) are used. MOM is formalized with the Electric Field Integral Equation (EFIE) for wires and the Magnetic Field Integral Equation (MFIE) for patches. The code employs both full matrix decomposition and Banded Matrix Iteration (BMI) solution techniques. The MOM, GTD and hybrid MOM/GTD techniques in the code are used to solve electrically small object problems, electrically

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered) large object problems and combination sized object problems.

Volume I of this report is the User Manual. The code execution requirements, input language and output are discussed.

Volume II is the Engineering Manual. The theory and engineering approximations implemented in the code are discussed. Modeling criterion are given.

Volume III is the Computer Code Documentation Manual. This manual contains extensive software information of the code.

FOREWORD

This document was prepared by The BDM Corporation, 1801 Randolph Road, S.E., Albuquerque, NM 87106, for Rome Air Development Center/RBCT, Griffiss AFB, NY 13441, under contract number F30602-81-C-0084. This is the Computer Code Documentation Manual for version 3 of the GEMACS computer code. The Engineering Manual and User Manual are submitted under separate covers. These reports are submitted in accordance with Item Number A003 in The Contract Data Requirements List. The program manager on this contract was Diana L. Kadlec. The principal investigator was Dr. Edgar L. Coffey. The GEMACS documentation editor was Mariellen Kuna.

This document contains a description of the GEMACS computer code. It is meant for computer programmers. The actual user of the code is referred to the GEMACS User Manual and Engineering Manual, RADC-TR- - , Volumes I and II, respectively.

This document contains an overall description of the code and a detailed description of the code's major modules and each subroutine within the various modules. It also contains a complete alphabetical index of each variable within the code as well as a detailed discussion of each of the named common blocks.

The basic document reflects the GEMACS code as of March 25, 1983. When major changes, or a significant number of minor changes, are to be incorporated into the code, change pages will be issued for user incorporation into the basic document. In addition, comment cards will be included within the code to indicate the basic date of the code and the date of the latest revised statements within the code.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Concluded)

LIST OF FIGURES

LIST OF FIGURES (Concluded)

GEMACS SUBROUTINES GLOSSARY

SUBROUTINE	FUNCTION
ASSIGN	ASSIGN SUBROUTINE NUMBER
BABS	ABSOLUTE VALUE
BACSUB	BACK SUBSTITUTION
BANDIT	BAND MATRIX
BEXP	EXPONENTIAL
BLKDAT	BLOCK DATA
BMIRHS	BANDED MATRIX ITERATION RIGHT-HAND SIDE
BTAN2	TANGENT OF 2 ARGUMENTS
BUBBLE	BUBBLE SORT
CABC	COMPUTE A, B, C COEFFICIENTS
CAPINT	END CAP INTERCEPT
CLSFIL	CLOSE FILE
CNTGND	CONNECTIONS TO GROUND
CNVGTD	CONVERT GTD ELEMENTS CONNECTED TO PLATES
CNVTST	CONVERGENCE TEST
CONJUG	CONJUGATE THE Z ARRAY
CONVRT	CONVERT OUTPUT
COORDS	COORDINATE TRANSFORMATIONS
CYAXIS	CYLINDER AXIS COORDINATE SYSTEM
CYLINT	CYLINDER INTERCEPT
CYLNDR	READ AND STORE CYLINDER GEOMETRY
DECOMP	DECOMPOSE MATRIX

SUBROUTINE	FUNCTION
DFPTCL	DIFFRACTION POINT ON CYLINDER RIM
DFPTWD	DIFFRACTION POINT ON WEDGE
DFRFPT	DIFFRACTION REFLECTION POINTS
DICOEF	DIFFRACTION COEFFICIENT
DIFPLT	DIFFRACTION BY PLATE
DMPDRV	DIRECT MANIPULATION DRIVER
DPI	WEDGE SLOPE DIFFRACTION COEFFICIENT
DPLRCL	DIFFRACTION BY PLATE, REFLECTION BY CYLINDER
DPLRPL	DIFFRACTION BY PLATE, REFLECTION BY PLATE
DPTNFW	DIFFRACTION POINT TO NEAR FIELD
DQG32	NUMERICAL INTEGRATION
DW	DIFFRACTION COEFFICIENT FOR WEDGE
DZCOEF	CURVED EDGE DIFFRACTION COEFFICIENT
EFDGEO	FIND EFIELD COMMAND GEOMETRY
EGFMAT	GENERATE E-FIELD GREEN'S FUNCTION MATRIX
ENDCAP	READ AND STORE END CAP GEOMETRY
ENDIF	END CAP RIM DIFFRACTION
ERROR	ERROR ROUTINE
ESPARM	DECODE ESRC COMMAND PARAMETERS
EXCDRV	EXCITATION DRIVER
FABLO2	OUTPUT ROUTINE
FABLO4	OUTPUT ROUTINE

GEMACS SUBROUTINES GLOSSARY (Continued)

SUBROUTINE	FUNCTION
FARFLD	FAR FIELD COMPUTATION
FCT	INTEGRAND FUNCTION
FFCT	TRANSITION FUNCTION
FKARG	DIFFRACTION ARGUMENT
FKY	TRANSITION FUNCTION FOR CURVED EDGE
FLDDRV	FIELD DRIVER ROUTINE
FLDOUT	FIELD OUTPUT ROUTINE
FLTPLT	TEST FOR FLAT PLATES
FNDARG	FIND ARGUMENT ROUTINE
FNDREC	FIND RECORD ON DATA FILE
FRNELS	FRESNEL INTEGRAL
FUNI	INTEGRAND FOR FKARG
GEMACS	GEMACS MAIN PROGRAM
GEODRV	GEOMETRY DRIVER
GEOM	GEOMETRIES OF PLATES
GEOMC	GEOMETRIES OF CYLINDER
GEOMPC	GEOMETRIES FOR PLATE-CYLINDER
GETARG	GET ARGUMENTS
GETFLD	GET FIELD POINT COORDINATES
GETGEO	GET PREVIOUS GEOMETRY
GETKWD	GET KEYWORD
GETKWV	GET KEYWORD VALUE

GEMACS SUBROUTINES GLOSSARY (Continued)

SUBROUTINE	FUNCTION
GETPNT	GET POINT DATA
GETSEG	GET SEGTBL RECORDS
GETSYM	GET SYMBOL DATA
GNDREF	GROUND REFLECTION
GTDCS	GTD COORDINATE SYSTEMS ROUTINE
GTDDRV	GEOMETRICAL THEORY OF DIFFRACTION DRIVER
IBITCK	INTEGER BIT CHECK
IMAGE	SOURCE IMAGE
IMCDIR	AXES IMAGE DIRECTION THRU END CAP
IMDIR	AXES IMAGE DIRECTION
INCFLD	INCIDENT FIELD
INPDRV	INPUT DRIVER
INTPLT	INTERPOLATE ELECTRIC FIELD (GTD)
JCTION	DETERMINE WIRE JUNCTIONS
JNCSUM	JUNCTION SUM
LITSCH	LITNUM ARRAY SEARCH
LNKGTD	LINK GTD GEOMETRY OBJECTS
LNKJCT	LINK SEGMENT JUNCTIONS
LODDRV	LOADS DRIVER
LODSYM	CHECK FOR LOAD SYMMETRY
LUDDRV	LOWER UPPER DECOMPOSITION DRIVER

GEMACS SUBROUTINES GLOSSARY (Continued)

SUBROUTINE	FUNCTION
LUSTAT	LOGICAL UNIT STATUS
MOVFIL	MOVE FILE
NANDB	CYLINDER NORMAL AND TANGENT
NERFLD	NEAR-FIELD COMPUTATION
NFD	NEAR-FIELD DISTANCE
NTGRAN	COMPUTE INTEGRAND
NTRPLT	INTERPOLATE ELECTRIC FIELD (WIRES)
NTRPLU	INTERPOLATE FIELDS (PATCHES)
OPNFIL	OPEN FILE
PAGPLT	PAGE PLOT
PARSE	PARSE
PATCH	READ AND STORE PATCH GEOMETRY
PFUN	p* FUNCTION
PLAINT	PLATE INTERCEPT
PLATE	READ AND STORE PLATE GEOMETRY
PLIST	PARSE LIST OF DATA
PLTSEG	DETERMINE IF PLATE IS CONNECTED TO A SEGMENT
POLYRT	POLYNOMIAL ROOTS
POSTIP	POST INPUT PROCESSOR
POSTPR	POST PARSE PROCESSOR
PREPAR	PRE PARSE PROCESSOR
PRESCN	PRE SCAN ARGUMENT LIST

GEMACS SUBROUTINES GLOSSARY (Continued)

SUBROUTINE	FUNCTION
PRTGTD	PRINT GTD GEOMETRY DATA
PRTKJ	PRINT KJ INTERACTIONS
PRTSYM	PRINT SYMBOL DATA
PUTKWV	PUT KEYWORD VALUE
PUTPNT	PUT POINT DATA
PUTSEG	PUT SEGMENT DATA
PUTSYM	PUT SYMBOL DATA
QFUN	q* FUNCTION
RADCV	RADIUS OF CURVATURE
RCLDPL	REFLECTION BY CYLINDER, DIFFRACTION BY PLATE
RCLRPL	REFLECTION BY CYLINDER, REFLECTION BY PLATE
RDEFIL	READ FILE
REBLCK	REBLOCK A SPECIFIED MATRIX
REFBP	REFLECTION INCIDENT DIRECTION
REFCAP	REFLECTION BY END CAP
REFCYL	REFLECTION BY CYLINDER
REFLCT	REFLECT GEOMETRY DATA
REFPLA	REFLECTION BY PLATE
RESTRT	CHECKPOINT RESTART
RFDFIN	CYLINDER REFLECTION POINT FOR NEAR FIELD
RFDFPT	REFLECTION, DIFFRACTION POINTS
RFPTCL	REFLECTION POINT ON CYLINDER

GEMACS SUBROUTINES GLOSSARY (Continued)

SUBROUTINE	FUNCTION
ROMBNT	ROMBERG INTEGRATION
ROTATE	GEOMETRY ROTATION ABOUT AN AXIS
ROTRAN	ROTATE TRANSLATE
RPLDPL	REFLECTION BY PLATE, DIFFRACTION BY PLATE
RPLRCL	REFLECTION BY PLATE, REFLECTION BY CYLINDER
RPLRPL	REFLECTION BY PLATE, REFLECTION BY PLATE
RPLSCL	REFLECTION BY PLATE, SCATTERING BY CYLINDER
RWCOMS	READ/WRITE COMMONS
RWFILS	READ/WRITE DATA FILES
SCALE2	SCALING LINEAR
SCALE3	SCALING LOGARITHMIC
SCAN	SCAN INPUT CARD
SCLRPL	SCATTERING BY CYLINDER, REFLECTION BY PLATE
SCTCYL	SCATTERING BY CYLINDER
SEJCON	SET J^{TH} CONNECTION DATA
SET	SET THE KJ INTERACTIONS
SETDRV	SET DRIVER
SHELL	SHELL SORT
SMAGNF	S MAGNITUDE IN NEAR FIELD
SMATRX	GENERATE SYMMETRICAL MATRIX OPERATOR
SOLDRV	SOLUTION DRIVER
SOLVIC	SOLVE IN CORE MATRICES

GEMACS SUBROUTINES GLOSSARY (Continued)

SUBROUTINE	FUNCTION
SOLVOC	SOLVE OUT OF CORE MATRICES
SOURCE	SOURCE FIELD PATTERN FACTOR
SOURCP	SOURCE SLOPE FIELD PATTERN FACTOR
SPWDRV	SPHERICAL WAVE DRIVER
STATFN	STATISTICS AT FINISH
STATIN	STATISTICS AT INPUT
STATOT	STATISTICS AT OUTPUT
STRTUP	START UP MODULE EXECUTION
SUBPAT	GENERATE SUBPATCHES (WIRE CONNECTED)
SYMDEF	SYMBOL DEFINITION
SYMLIT	SYMBOL/LITERAL SEARCH
SYMMOD	SYMMETRICAL MODIFICATION OF Z ARRAY
SYMSCH	SYMBOL TABLE SEARCH
SYMUPD	SYMBOL UPDATE
SYSCHK	SYSTEM CHECK
SYSRTN	SYSTEM ROUTINES
TANG	TANGENTS
TICHEK	TIME CHECK
TNEFLD	TANGENTIAL ELECTRIC FIELD
TNHFLD	TANGENTIAL MAGNETIC (H) FIELD
TPNFLD	THETA, PHI COMPONENTS IN NEAR FIELD
TRCEBK	TRACE BACK

GEMACS SUBROUTINES GLOSSARY (Concluded)

SUBROUTINE	FUNCTION
TRNLAT	POINT TRANSLATION
TSKXQT	TASK EXECUTION
UNEFLD	UNIT CURRENT ELECTRIC FIELD
UNHFLD	UNIT CURRENT MAGNETIC (H) FIELD
WLKBCK	WALK BACK
WRTCHK	WRITE CHECKPOINT
WRTFIL	WRITE TO FILE
WYRDRV	WIRE INPUT DRIVER
WYRPAT	WIRE FIELD DUE TO PATCH
XYZFLD	X,Y,Z COMPONENTS IN NEAR FIELD
ZCDRVR	Z CODES DRIVER
ZGTDRV	Z MATRIX GTD DRIVER
ZIJDRV	ZIJ DRIVER
ZIJSET	ZIJ SET
ZINT	Z INTERNAL OF CIRCULAR WIRE
ZZXDUM	ZZX DUMMY SUBROUTINE CALL

GEMACS MAJOR ARRAYS
GLOSSARY

ARRAY NAME	CONTENTS
CVAL	COORDINATE SYSTEM DATA
FLTSYM	INTERNAL SYMBOL STORAGE TABLE
IDEFIN	DEFINED ELEMENT DATA
INTARG	TEMPORARY STORAGE OF NARGTB DATA
IOFILE	CURRENT FILE POSITION
ISHADW	SHADOWING MATRIX
KJINT	INTERACTION ARRAY
KWARG	KEYWORD ARGUMENT TABLE
KWFMTP	KEYWORD FORMAT TABLE POINTER
KWNAME	KEYWORD NAME TABLE - POINTS TO NCODES TABLE
LITNUM	LITERAL STORAGE TABLE
NARGTB	ARGUMENT LIST DATA
NCODE	INPUT FIELD CODE TABLE
NCODES	INTERNAL CODED BCD
NDATBL	SYMBOL DATA TABLE
NDFILE	NUMBER OF WORDS ON FILE
NLOOPS	LOOP CONTROL TABLE
NTSFMT	TASK FORMAT TABLE
NTSKTB	TASK TABLE - POINTERS TO NARGTB
NVAL	INPUT FIELD VALUE TABLE
PTTBLE	POINT COORDINATE TABLE
SEGTBL	SEGMENT DATA TABLE
TEMP	SCRATCH STORAGE

GEMACS NAMED COMMONS
GLOSSARY

COMMON BLOCK	USE
ADEBUG	GLOBAL DATA
AMPZIJ	INTERACTION MATRIX GENERATION DATA
ANUM	SEGMENT LENGTH INTERPOLATION DATA AT JUNCTIONS
ARGCOM	ARGUMENT LIST DATA
CSYSTM	COORDINATE SYSTEM DATA
CYLIN	INPUT CYLINDER GEOMETRY
DEFDAT	DEFINED ELEMENT DATA
FLDCOM	FIELD OUTPUT DATA
FLDVAL	E-FIELD COMMAND LOOP AND COORDINATE DATA
FLDXYZ	TOTAL ELECTRIC FIELD
GEODAT	GEOMETRY DATA
GEOMEL	CYLINDER GEOMETRY IN WAVELENGTHS
GEOPLA	PLATE GEOMETRY IN WAVELENGTHS
GTDDAT	GTD GEOMETRY DATA
INPERR	INPUT ERROR DATA
INTMAT	INTERACTION DATA
IOFLES	I/O FILES DATA
JUNCOM	JUNCTION DATA
MODULE	MODULE NAME AND STATUS
PARTAB	PARSE TABLES
PLAIN	INPUT PLATE GEOMETRY
PNTTBL	POINT TABLE
ROTRDT	ROTATE, TRANSLATE DATA

GEMACS NAMED COMMONS

GLOSSARY (Concluded)

COMMON BLOCK	USE
SCNPAR	SCAN/PARSE DATA
SEGMNT	MOM SEGMENT DATA
SYMSTR	SYMBOL STORAGE DATA
SYSFIL	SYSTEM FILES DATA
TEMP01	SCRATCH STORAGE AREA
TMI	INTERACTION CALCULATION PARAMETERS

CHAPTER I
PROGRAM DESCRIPTION

A. OVERVIEW OF THE GEMACS CODE

The GEMACS program is a modularized, task-oriented code. Four modules are present in version 3 of GEMACS:

(a) INPUT--performs all input processing of commands and geometry.

(b) GTD--performs all calculations required for geometrical theory of diffraction (GTD) modeling.

(c) MOM--performs all calculations required for method of moments (MOM) modeling and numerical solutions of the resulting matrix problem.

(d) OUTPUT--prints and plots electric field patterns for GTD, MOM, or MOM/GTD hybrid solutions.

The basic structure of the four modules is similar and is presented in figures I-1 through I-4. The executive routines initialize the processing and interface with the host system to obtain various information used throughout the run. As a minimum, the system information required is the time of day or elapsed time since start of processing.

The four modules execute sequentially, as shown in figure I-5. However, should the functions of a module not be required in problem solution, that module may be bypassed. For instance, a MOM-only problem would not require the GTD module, so only the INPUT, MOM, and possibly OUTPUT modules would be needed.

The only communications medium among the modules is the checkpoint file. This file contains the contents of important commons and the contents of all data files present at the end of a module's execution. This allows two types of runs to be made. One may either execute all four modules in one batch job, or one may execute the modules in successive jobs, examining all intermediate results before proceeding to the next module. If the latter is the case, the checkpoint file should be a permanent file so that the checkpoint will be available for the next batch run.

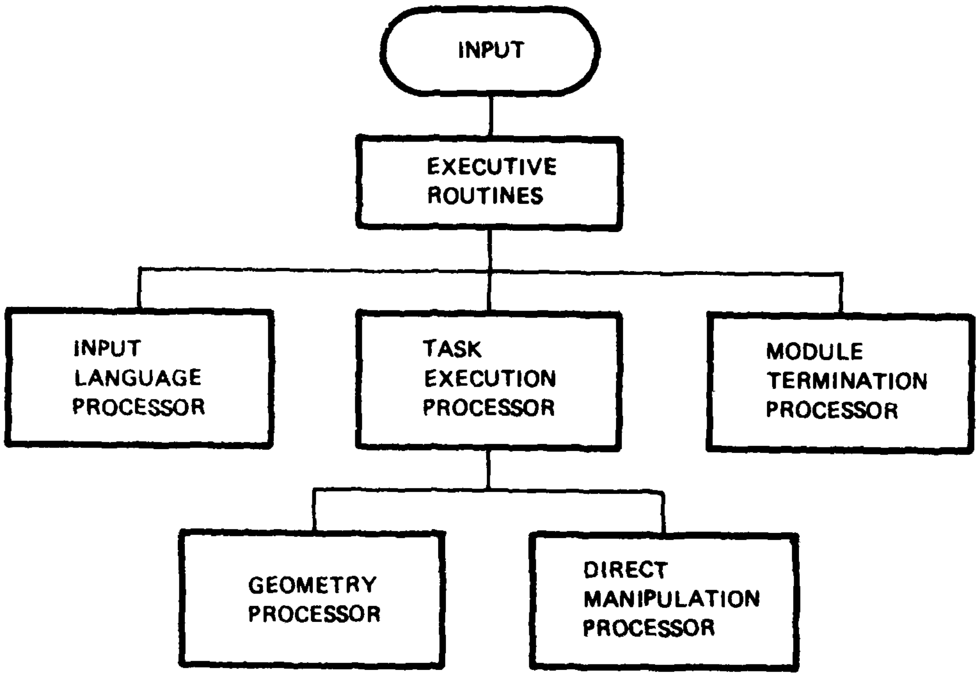


Figure I-1. Computer Code Structure for the GEMACS INPUT Module

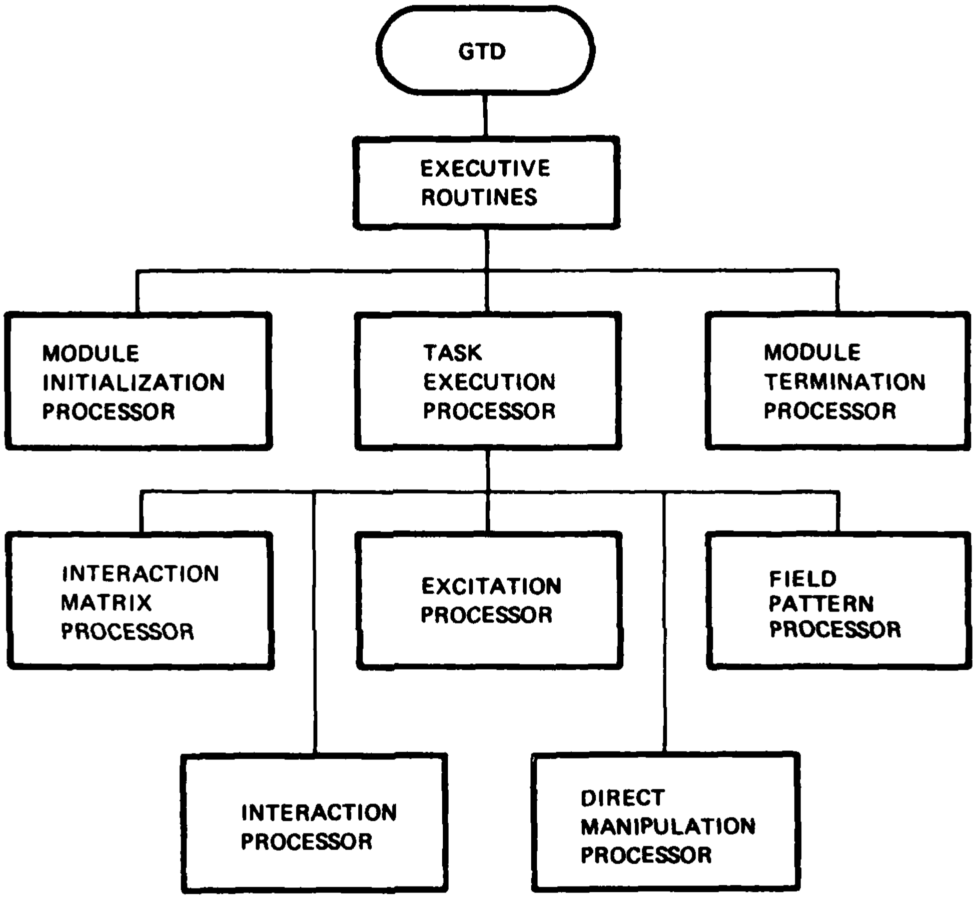


Figure I-2. Computer Code Structure for the GEMACS GTD Module

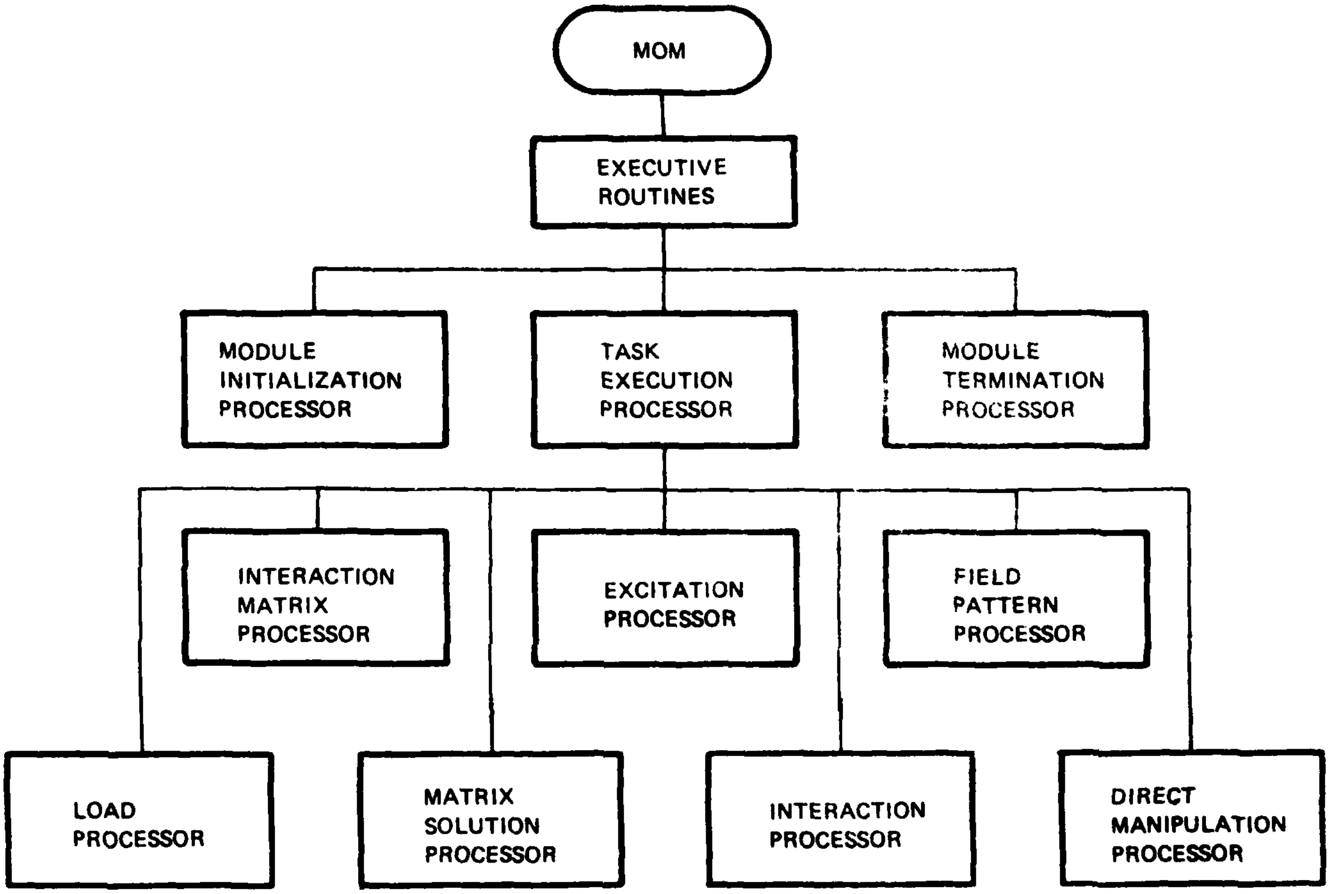


Figure 1-3. Computer Code Structure for the GEMACS MOM Module

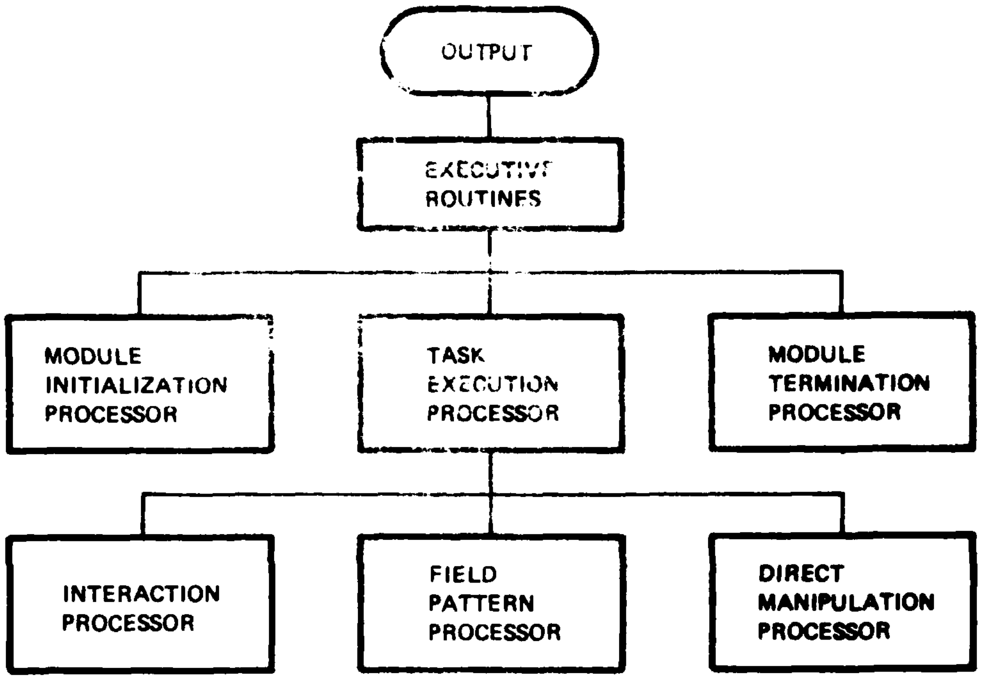


Figure I-4. Computer Code Structure for the GEMACS OUTPUT Module

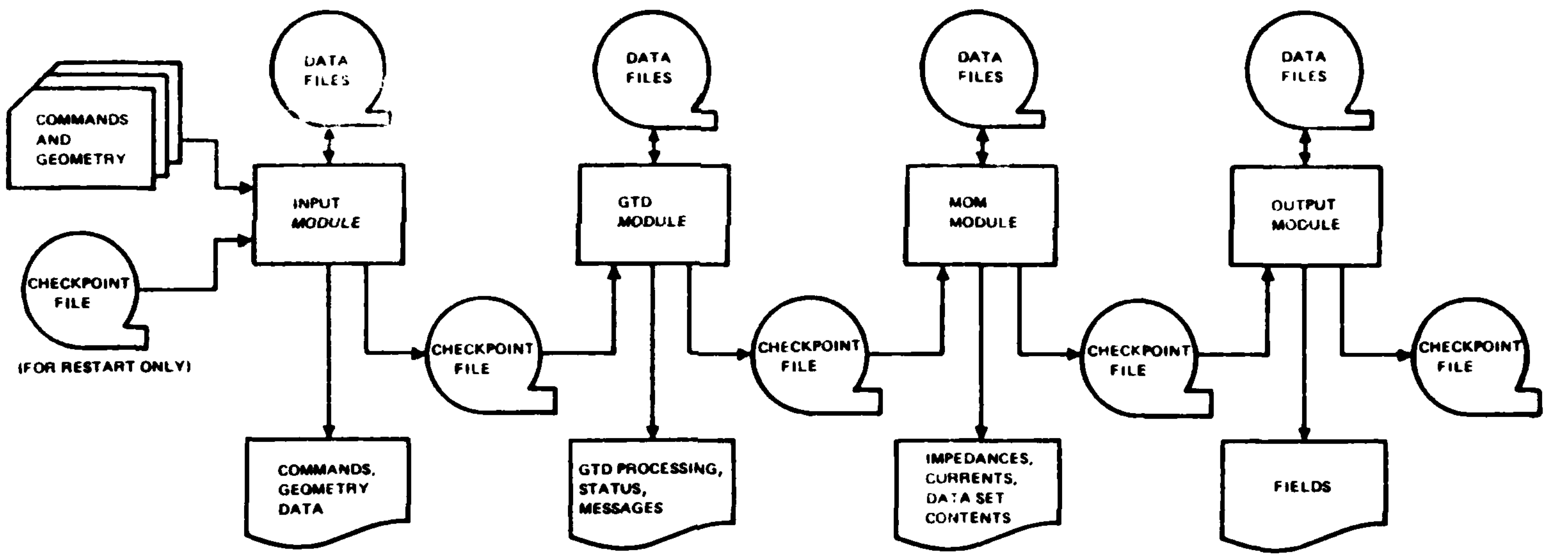


Figure I-5. Communications Among the Four GEMACS Modules

Processing always begins with the INPUT module (figure I-1). In the INPUT module, the Input Language Processor is called to read the user command language input until an END card is encountered. Control is then returned to the executive routines, and transfer is made to the Task Execution Processor which interprets the task list generated by the Input Language Processor and calls the appropriate subroutines to perform the tasks specified. There are two passes through the task list. During pass one, checks are made to ensure correctness of the user's input, establish symbol attributes, and establish those peripheral files needed for execution. After pass one, the symbols generated by the Input Language Processor are assigned storage. Single variables are stored in core on a first-encountered-first-stored-as-available basis. When internal core is exhausted, the symbols are assigned storage on the next logical unit available. These are sequentially accessed files. Since this is a very severe restriction, storage is often reassigned during execution. When this occurs, logical unit assignment starts with file 8 and proceeds sequentially until the available units are exhausted. When a symbol is purged, its logical unit becomes available for other symbols. The purged symbol's unit is closed by writing an end-of-file and rewinding back over the end-of-file.

During pass two, the user's commands are executed. Each command is executed sequentially by a subroutine designed to interpret the argument list specified for the command.

The INPUT module itself executes only geometry and direct manipulation commands. However, since the symbol attributes and task tables are stored in named common blocks, they are available to all subsequent modules, and no input processing is required by other modules.

The data interfaces in all GEMACS modules are primarily through the various data arrays located in named common blocks. Figure I-6 illustrates the use of the various arrays by function. The arrays used as input for the SCAN function are preloaded in the block data subroutine, BLKDAT. The SCAN function generates arrays used as input for the PARSE function. Additional PARSE function input is preloaded in BLKDAT. The

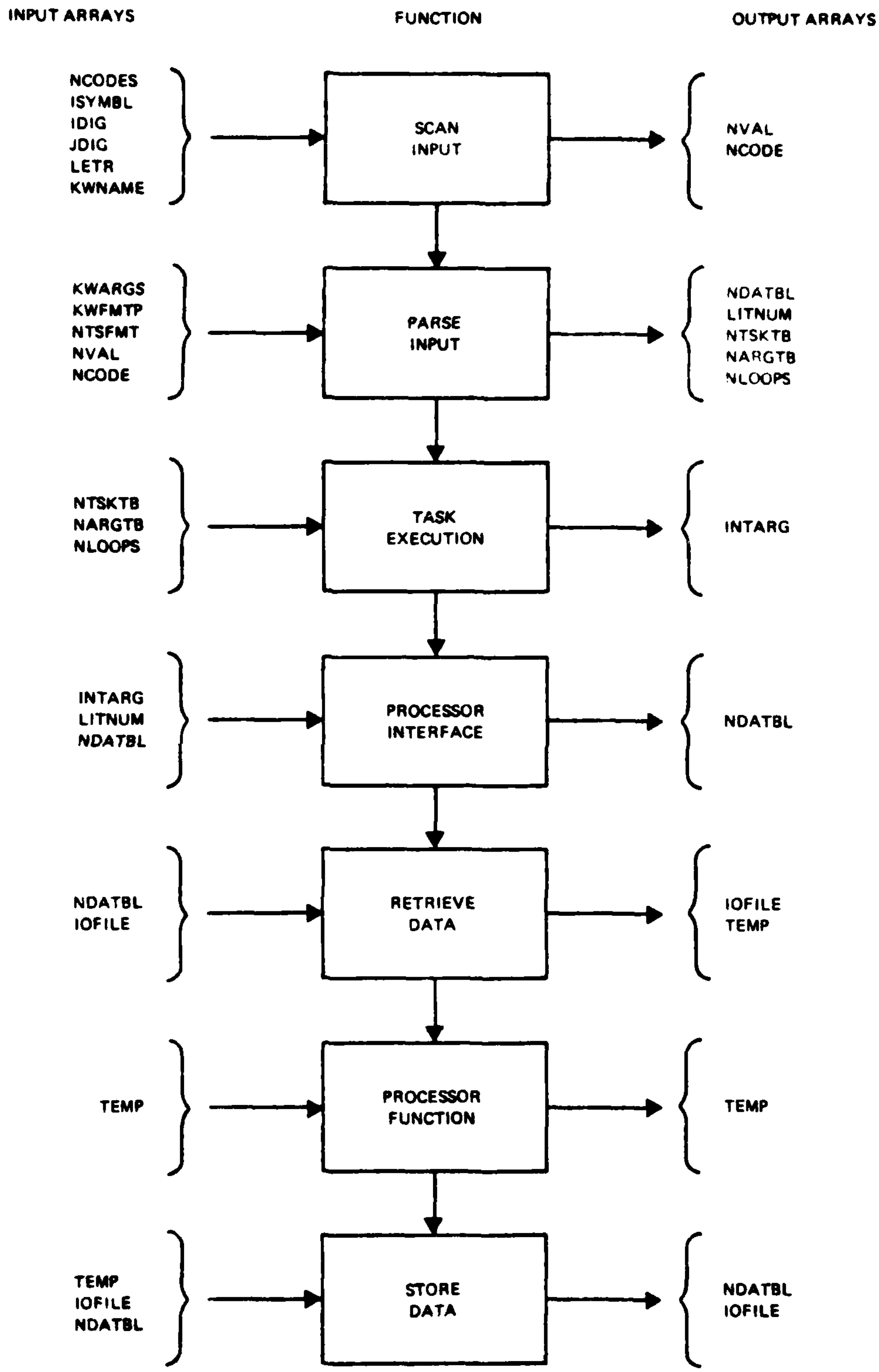


Figure I-6. Array Interface During Program Execution parsing function [illegible] literal and symbolic data in addition to providing the data needed [illegible]

The task [illegible] data from the NARGTB array and transfers it to the INTARG array for use by the task processor interface routines. This transfer of data is done to preserve the integrity of the argument list data [illegible] for subsequent modules, since the task interface processors often modify the argument list data.

The primary function of the processor interface is to initialize the argument list data (array INTARG), set up the data retrieval (before task execution), and store the data (after the task is complete). In doing this, the [illegible] array will be modified.

The data retrieval will utilize the contents of the IOFILE and NDATBL arrays to retrieve the contents of the data sets from the peripheral files. The TEMP array is used primarily for the internal storage of these data. There are exceptions to this, and they are pointed out in the description of the interface processors where they occur.

After the data are in core, the mathematical and other operations constituting the functions are performed. The data are typically overwritten in the TEMP array.

Upon completion of the operations, the modified data are returned to the peripheral file by the store data function. Again, the NDATBL and IOFILE arrays are used to control this process. After the data are stored, the TEMP array is available for use by the next processor. This logic and data flow continue until the module execution is complete.

At its termination each module stores all important named common blocks and all data files in a single checkpoint file. This file is used by the next module in sequence to initialize its named common blocks and data sets. In the GTD, MOM, and OUTPUT modules, scan and parse processing are not required. Instead, a processor is called to initialize the module with checkpoint file data. Then task execution begins with its first task in the task table and proceeds as discussed above. However, since different tasks are active in different modules, different results are obtained. The logic and data flow of figure 1-6 continue until module execution is complete.

Upon execution of all modules, the desired results are available in the data files. Some results are available prior to completion of all modules, as shown in figure I-5.

B. EXECUTIVE ROUTINES

The GEMACS executive routines control the interface of the computer code with the host computer. These routines also call the processors shown in table I-1. Those subroutines that are required by all top-level processors are in the executive processor. There are four primary functions performed by the executive routines. These are:

(a) Initializing or restarting a module
(b) Communicating with peripheral files
(c) Checkpointing
(d) Accumulating statistical information.

TABLE I-1. TOP LEVEL GEMACS PROCESSORS FOR EACH MODULE

	TOP-LEVEL PROCESSOR			
MODULE NAME	INPUT LANGUAGE	MODULE INITIALIZATION	TASK EXECUTION	MODULE TERMINATION
INPUT	X		X	X
GTD		X	X	X
MOM		X	X	X
OUTPUT		X	X	X

1. Module Restart or Initialization

The checkpoint file contains a snapshot of the code status at the time the checkpoint was written. This information is required to initialize a GTD, MOM, or OUTPUT module with the status and data of the previous module at its termination. Or, the data contained in an intermediate checkpoint can be used to restart a module at the point where the checkpoint was taken. The first case occurs in normal module flow. The latter option is used when an error occurred, when time ran out prior to finishing execution, or when the user wishes to restart with a modified command stream.

Module initialization is accomplished by subroutine STRTUP in conjunction with subroutines RWCOMS, RWFILS, PUTSYM, and GETSYM. The MODCHK file is searched for the last checkpoint written. Named common blocks are restored with data from this checkpoint by RWCOMS after which all peripheral files are reinitialized with data by RWFILS. The files are rewound, interfaces cleared, and control variables initialized. Then control is turned over to the task execution processor. If an error occurred in a previous module, execution is terminated at this point. Otherwise, normal task execution begins again at task 1 and proceeds until an END command is encountered.

A restart is the result of the input processor interpreting a RSTART command. Restart occurs in two steps. First, the input processor calls RESTRT (INPUT module) to initialize the INPUT module named common blocks and data files with data from the checkpoint file IOCKPT. This step destroys any input already processed by the input processor. However, upon completion of the restart initialization, additional commands may be added or original commands deleted with normal input processing functions.

The second step in restarting requires selecting the module in which to restart. (The module name is specified on the RSTART card.) The INPUT module is the first to check for a match. If a match is found, execution is turned over to the task execution processor. Execution in this case does not begin with the first task. Instead, execution is started with the task which wrote the checkpoint from which the restart was taken. An important exception to this rule occurs if the checkpoint was written at the termination of the OUTPUT module. Then, task execution begins with the first new task input after the restart.

If the INPUT module was not the module specified on the RSTART card, an end-of-module checkpoint is written, and the next module begins with subroutine STRTUP initializing the module as discussed above. If STRTUP detects that this initialization is a part of a restart, module names are compared. If a match is found, execution is turned over to the task execution processor of this module. Otherwise, this module stops executing so that the next module (in sequence) may compare module names. This continues until a match is found or all modules are exhausted.

2. Peripheral File I/O

The I/O is accomplished through four subroutines. Subroutines GETSYM and PUTSYM are called to fetch and store the data associated with the symbols defined by the user and internally defined symbols. These subroutines, in turn, call subroutines RDEFIL and WRTFIL for all actual I/O to the peripheral devices. There are two exceptions to this rule in the GEMACS code. These occur in MOM subroutines DECOMP and SOLDRV. These subroutines interface directly with the subroutines RDEFIL and WRTFIL without going through subroutines GETSYM and PUTSYM. The motivation for this direct interface is the savings in time realized by not calling subroutines GETSYM and PUTSYM with alternating symbol names.

Subroutines GETSYM and PUTSYM determine the record length for each data set from the attributes associated with the data set. They then determine the correct file and the starting position of the requested record on that file. The file is then positioned at the starting location of the correct record by subroutine MOVFIL. The data are retrieved or stored using subroutines RDEFIL or WRTFIL as required. Due to the ANSI FORTRAN restriction on the GEMACS code, all files are sequential.

In order to minimize the processing time associated with file manipulation, there is an external file required for each data set which will have a dimension greater than 1. The user must make the logical units available to the GEMACS code to store all the symbols requiring peripheral file storage. These units start at logical unit number 8 and go as high as the user desires. The highest numbered logical unit should be input to GEMACS using the NUMFIL keyword entry in the command language. It is explicitly assumed that all logical units between 8 and that specified on the NUMFIL entry are available for use by GEMACS.

The total number of logical units required may be reduced by the use of the PURGE command when the actual data associated with a symbolic name are no longer required. The symbol name may still be referenced in a subsequent command. A typical example of this occurs when a matrix has been decomposed into its lower and upper triangular components; the interaction matrix may be purged if it is not going to be used later. Reference to the interaction matrix in the BACSUB command may still be made; the lower and upper triangular matrices associated with the purged matrix will be retrieved. In a similar manner, the lower and upper triangular matrices resulting from a decomposition may be purged after the solution vector has been found. This will make more storage available for computation of such quantities as the near and far electric fields if the user desires to save the data associated with these quantities.

A pseudorandom access capability is contained in the GEMACS code. This means that records contained within a data set and elements of these records may be replaced. This is accomplished by using two files: a scratch file to store the original data up to the record being modified, the record being modified is then written out onto the scratch file being used, then the remainder of the data associated with the symbol is transferred to the scratch file. Once all data are on the scratch file, it is rewound and transferred back to the original file. For large data sets, this is a very expensive and time consuming operation and should, in general, be avoided unless true random access capability has been implemented in the GEMACS code.

The looping capability and multimodule organization of GEMACS make it necessary to store intermediate results for all data sets. For instance, the generation of a MOM/GTD hybrid interaction matrix at six frequencies requires that the matrix for each frequency be retained throughout program execution, because the data at all six frequencies must be available to the MOM module as well as the GTD module. Were GEMACS to write over intermediate data in the GTD module as the frequency was incremented, these data would be lost to the MOM module. Therefore, unlike the standard FORTRAN convention of data replacement, GEMACS uses the concept of data concatenation for peripheral files. Data for scalar variables are still replaced, since these data are generated anew in each module by the direct manipulation processor.

3. Checkpointing

Checkpointing is accomplished by subroutine WRTCHK in conjunction with subroutines RWCOMS, RWFILS, PUTSYM, and GETSYM. The checkpointing feature of GEMACS is implemented by writing all of the data stored in named common blocks to a peripheral file. After the named common blocks have been written to the checkpoint file, the symbol table is scanned, and all symbols having data stored on a peripheral file will have those data retrieved and rewritten onto the checkpoint file.

Two checkpoint files are available: MODCHK (for end-of-module checkpoints), and IOCKPT (for timed and command checkpoints). MODCHK and IOCKPT are both initialized to FORTRAN logical unit 7. This causes all checkpoints to be written to the same file. However, upon successful execution of a module, MODCHK is rewound prior to writing the end-of-module checkpoint. To save the timed and command checkpoints, it is necessary to specify a different logical unit number for IOCKPT in the FILEID item of the CHKPNT command.

An optional item, NR, on the checkpoint command has been provided to inhibit rewinding of the IOCKPT checkpoint file after each checkpoint. If this parameter is not specified, the checkpoint file will have a FORTRAN END OF FILE written, and a rewind command will be issued to the file. The next checkpoint taken will then overwrite the previous checkpoint. If the NR item is specified, the checkpoint file will not be rewound nor will there be an end-of-file mark written on the checkpoint file. Subsequent checkpoints will be written in a contiguous fashion on the checkpoint file. The user may then recover at any checkpoint that is present on the IOCKPT file by specifying the checkpoint number in the CPNUM item of the RSTART command.

A checkpoint file may become quite large during the solution of a large problem. For this reason, care should be taken when the NR parameter is specified.

4. Accumulating Statistical Information

During the execution of each module of the GEMACS code, statistical information relating to the number of times each subroutine is entered and the total time in CP seconds spent in each subroutine may be accumulated. This option is selected by the DEBUG STATS command. This information provides the user with some insight into the source of the expenses incurred during program execution. This information is printed at the conclusion of the execution. Using this information, the user may determine those portions of the code which may require optimization for a particular use.

C. GEMACS INPUT PROCESSOR

1. Input Language Processor

The basic function of the Input Language Processor is to translate the user's free-field input into a task list table in order that the execution processor may complete the user's commands. All input processing is performed by the INPUT module.

Figure I-7 illustrates the logical and functional subroutine interfaces in the Input Language Processor. Identified in the figure are the major functions and the subroutines that perform these functions. Detailed subroutine flowcharts are presented in chapter II.

The basic array used by the Input Language Processor is the NCODES array. This array, loaded in the BLKDAT subroutine, contains all of the internal code representation for the keywords and other text used by the GEMACS code.

The user's input strings are scanned by the subroutine SCAN and the individual fields which are delimited by blanks, commas, parentheses, or arithmetic operators are extracted from the input command. Scanning will continue for all fields related to the same command before returning to the calling subroutine. Continuation on an input command is indicated by the presence of a continuation character (set in variable NCONCH in common SCNPAR in subroutine BLKDAT) in column 1 of each succeeding record of the command.

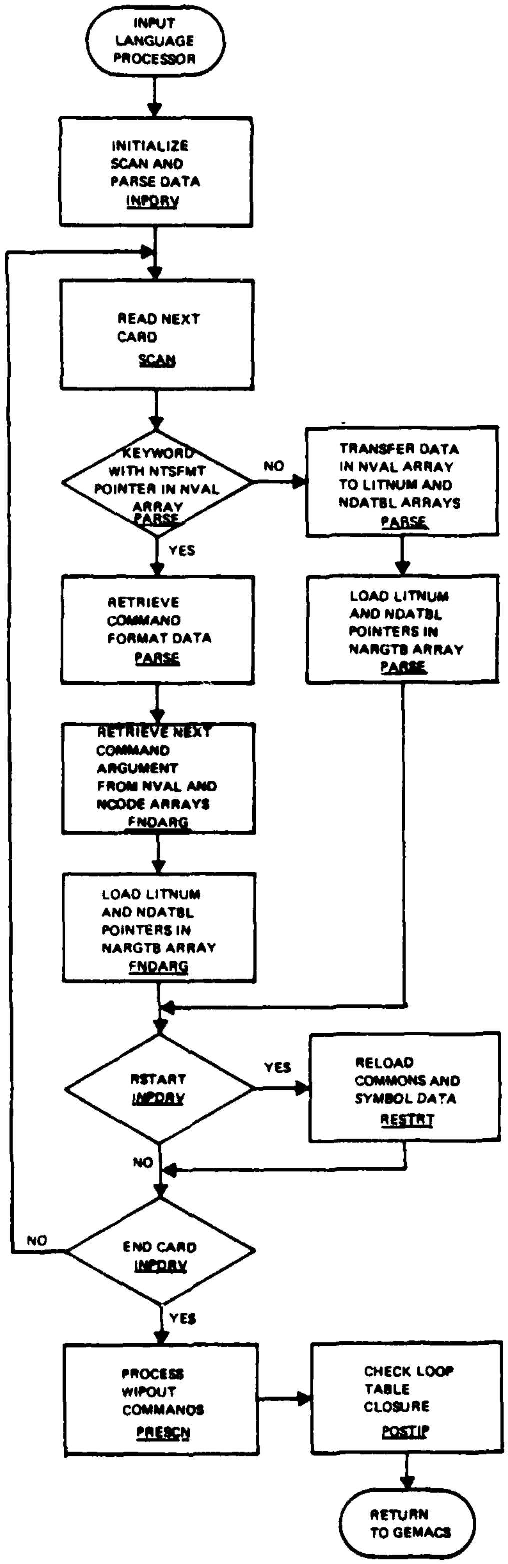


Figure I-7. Input Language Processor Function/
Subroutine Interface

The rules for field construction are as follow:

(a) All alphanumeric input must begin with an alpha character.

(b) All numeric input begins with a numeric character or decimal point.

An error will occur if a field beginning with a numeric character subsequently is found to contain alpha characters with the exception of a floating point field which may contain the alpha character 'E' to indicate the exponent input mode for numeric fields. Special characters such as the characters associated with arithmetic operations and parentheses constitute a field by themselves.

The output of the SCAN subroutine is contained in two arrays, the NVAL and NCODE arrays. The NVAL array contains the values of the fields while the NCODE array contains a code indicating what type of field is present. The types of fields recognized are:

(a) A symbol or special character field indicated by the value of NTSYMB

(b) A keyword field indicated by the value of NTKEYW

(c) An alphanumeric field indicated by the value NTALPH

(d) An integer field indicated by the value NTINT

(e) A floating point field indicated by the value NTFLPT

(f) The END card indicated by the value NTEND.

The values associated with these symbols are loaded in subroutine BLKDAT initially; and when a field corresponding to one of these types is detected by the SCAN routine, this corresponding value is loaded into the NCODE array at the proper location for the field being scanned. The NVAL array and its equivalenced VAL array contain the actual value from the user's input for each field encountered. If a keyword is encountered in the input stream, the user's input is replaced by the index of the keyword in the array KWNAME. If a symbol such as an asterisk or slash is found, the symbol is replaced by an integer value corresponding to the location of the symbol in the ISYMBL array.

Once the scanning is complete, control is returned to the INPDRV subroutine for a subsequent call to the PARSE subroutine. When control is transferred to the PARSE subroutine, the NVAL and NCODE arrays are searched for keywords. If no keywords are found, it is assumed that the command is a direct manipulation command such as those represented by arithmetic operations on symbols, and the entire contents of the NVAL array for the previous command entry are loaded directly into the task table to be processed by subroutine DMPDRV. If a keyword is found, the array KWFMTP is searched for a nonzero entry corresponding to the keyword found. A nonzero entry in this array points to the location in the NTSFMT array which contains the format for the task associated with this keyword. If a zero entry is found in the KWFMTP array, then there is no task associated with the current keyword, and the search is continued through the NCODE array until a keyword with a nonzero entry in the KWFMTP array is found in order to determine the task being processed. As stated previously, if no nonzero entry is found for the keyword in the present input command, it is assumed to be a direct manipulation command and will be processed by the direct manipulation processor DMPDRV.

The first keyword found which has a task associated with it will determine how the task is interpreted and processed. Once a nonzero entry is found for KWFMTP, the data starting at that position of the NTSFMT array are retrieved. The first entry at that position is the number of entries which follow pertaining to the task associated with the keyword found. The remaining entries in the NTSFMT array determine the format of the command. The first entry after the number of entries indicator contains the task number associated with the keyword. This task number is loaded into the next available position of the task execution list which is stored in the array NARGTB. The location in this array of the task number is stored in the array NTSKTB. The subsequent entries in the NTSFMT table for a given command are coded to indicate either keywords, alphanumerics, literals, or a list of these types of arguments to be searched for in the input stream. If any entry required by the NTSFMT array is not present, a NOPCOD value is loaded into the corresponding position of the NARGTB array. During pass 1 of execution, the subroutines required to execute the command will determine if there is adequate information. In this way, the first pass of execution acts as a second pass of the parser. If information is not provided by the user, the default value will be used if it is available; if there is no default parameter, or no default value provided for a given parameter, an error will occur during pass 1. All remaining commands will be scanned during pass 1; however, actual execution which takes place during pass 2 will be inhibited.

For a given argument type found in the NTSFMT array for the command being parsed, subroutine FNDARG is called to locate either the argument or the value associated with a keyword argument. Subroutine FNDARG will call the subroutines SYMLIT, LITSCH, or SYMSCH in order to determine the location of symbols or literals previously entered via preceding commands. Symbolic names defined by the user are stored in the data table NDATBL. Associated with each NDATBL entry is a list of attributes associated with the symbols. These attributes are determined during pass 1 and pass 2 of execution, not during input language processing. Numeric fields encountered, and also alpha fields encountered on the right-hand side of an equal sign, are stored in the literal table LITNUM. In order to store floating point numbers, the array FLTLIT is equivalenced to the literal table array LITNUM. There are two entries for each value stored in the literal table. These are:

(a) The literal code (KOLCOD)

(b) The literal value (KOLVAL).

The literal codes are the values associated with the parameters NTKEYW, NTALPH, NTINT, and NTFLPT. In this way, subsequent use of a particular entry in the literal table may determine the nature or the type of entry present.

When FNDARG creates an entry in the NDATBL array, a positive integer pointing to that location is loaded into the NARGTB array for task execution. When an entry is made into the LITNUM array, a negative integer, whose absolute value is the position in the LITNUM array of the entry, is loaded into the NARGTB array. The NARGTB array, which is used as the task execution array, contains positive integers pointing to the

NDATBL array and negative integers pointing to the LITNUM array. The order of the integers is determined by the NTSFMT entries. In this way, the free-field, unordered input of the user is interpreted and ordered before it is stored in the NARGTB array for subsequent execution.

The Input Language Processor continues in this manner for each command input by the user until a command is encountered which contains an END starting in column 1. The END card may contain subsequent items to indicate to the user, for his convenience, the end of the input for a particular group of commands.

In the event that a restart from checkpoint command has been encountered, control will be transferred to the RESTRT subroutine to restart from checkpoint. If this occurs, subsequent commands are added on to the end of the command stream present for the run which generated the checkpoint. After the checkpoint restart has been accomplished, control is returned to the INPDRV subroutine and subroutine SCAN is reentered to continued processing the user's current input stream.

Upon encountering an END card, subroutine PRESCN is called to eliminate those NTSKTB and NARGTB entries removed by the WIPOUT command.

Subroutine POSTIP will be called to verify that the loop table NLOOPS is correctly loaded (i.e., no open loops). Additionally, if the debug mode for the ILP was requested, subroutine POSTIP will list the contents of the NTSKTB, NARGTB, NDATBL, LITNUM, and NLOOPS arrays.

2. Addition and Modifications of GEMACS Commands

The process of adding commands to GEMACS involves making entries into the NCODES array, the KWNAME array, the KWARG array, the NAMTSK array, the KWFMTP array, and the NTSFMT array. In addition, an entry starting with the letters KW with up to four subsequent characters indicating the keyword mnemonic must be made in the section in subroutine BLKDAT labeled KEYWORD INDICES. All entries must be composed of the characters contained in the ANSI FORTRAN character set and no others are allowed. This is in order to preserve the basic BCD nature of the GEMACS code for internal representation of BCD strings.

The addition process, illustrated in figure I-8, is discussed in detail below. The entries in the NCODES array are constructed in the following manner: the first character of the BCD string is located as an entry in the first 45 positions of the NCODES array. The entry in this array represents the internally coded value for the BCD character, for instance, A is represented as a 1, B as a 2, and so forth. The second character of the BCD string is also located in the first 45 positions of the NCODES array. The internal representation of the BCD string is then built by multiplying the internal representation of the first character by 2^6 and adding the representation of the second character. The internal representation of the third character is then found in the NCODES array and the subsequent representation of the first three characters of the string is determined by multiplying the representation of the first two characters by 2^6 and adding the internal representation of the third character. This process continues until all characters of the string, maximum of six, have been represented by their internal representation. The integer resulting from this operation is then loaded into the next available, or any available, location of the NCODES array.

The next task is to make an entry in the KWNAME array. This entry is a pointer to the location of the internal representation of the BCD string in the NCODES array. When this string is encountered in the user's input stream, the string will be replaced by the number of the location in the KWNAME array pointing to the location in the NCODES array which matches the string encountered.

The next task is to make a corresponding entry in the KWFMTP array. "Corresponding" means that the KWNAME array and the KWFMTP array have the same number of entries, and the same positions of each array refer to the same keyword. If there is to be a task associated with this keyword, or if this keyword is to be used to identify a particular command, the entry in the KWFMTP array must point to the start location in the NTSFMT array which contains the coded format to be used to decode this command during parsing. In addition, an entry must be made in the NTSFMT array starting at the location specified in the KWFMTP array.

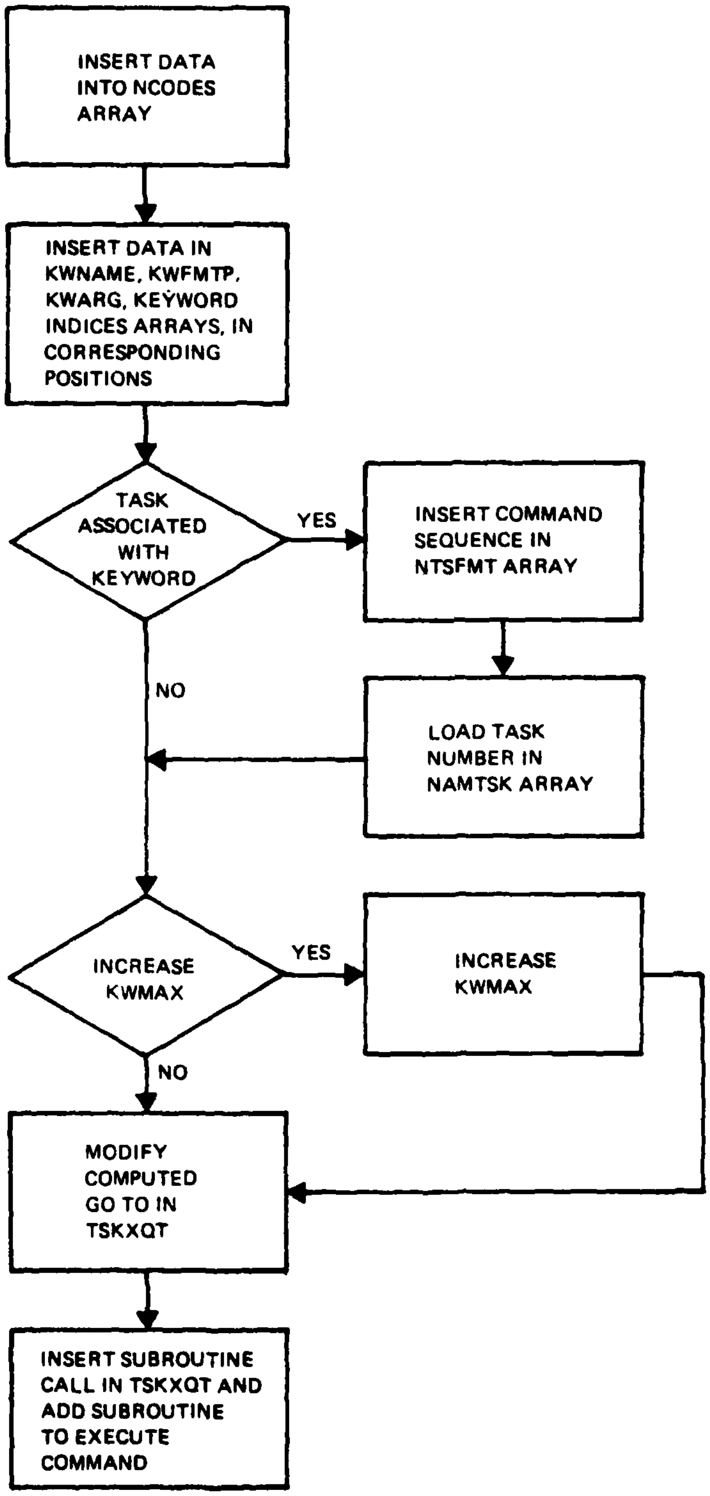


Figure I-8. Command Addition Flowchart

This entry is the coded representation of the command using the codes documented in the BLKDAT subroutine. Entries may be made at any available location which has a sufficient number of contiguous nonused words to accommodate the command being entered. It is important to remember that the first entry in NTSFMT for a keyword which is to have a task associated with it is the number of subsequent entries associated with that task. The following entry is the task identification number, and it must be a unique number for the given task since this number will be extracted from the NARGTB array by the task execution processor and used in a computed GO TO to call the subroutine necessary to execute the task. This task identification number is loaded at the end of the array NAMTSK, and its value acts as a pointer to the NCODES array, identifying the location of the name of the task in array NCODES. In addition to the KWFMTP array entry, an entry must be made in the KWARG array; again this array must have entries in the same position as the KWNAME and the KWFMTP arrays. The entry in the proper position in the KWARG array indicates the type of argument which this keyword may have. A zero entry implies no argument for this keyword. The argument types are documented in the block data subroutine BLKDAT.

Finally, an entry must be made in the area labeled KEYWORD INDICES in subroutine BLKDAT. Its position must correspond to the position of the data referring to the keyword in arrays KWNAME, KWFMTP, and KWARG. The symbol is made up of six or less characters, the first two of which are KW. The remaining characters would be a mnemonic relating to the keyword.

Note that the same keyword may be used in more than one command; and it is important to realize that if more than one task-oriented keyword occurs in a command, the first occurrence of such a keyword in the command will determine the task identification. Therefore, if a command contains two keywords, the order of occurrence in the command is important. The keyword identifying the task <u>must</u> occur first in the user command input.

If the number of entries in the labeled KEYWORD INDICES section of subroutine BLKDAT is increased, the variable KWMAX in BLKDAT must be updated to reflect the new total number of entries in the area. Under no circumstance is KWMAX to exceed 255, the value of the variable KWLMT. If the number of keywords does exceed 255, then KWLMT must be increased to the next power of 2, that is to 512. If this is done, then the packing format for keywords must be restructured. The variable MKMX in BLKDAT must be changed from 3 to 2, indicating a maximum of two keywords that may be packed in a multiple keyword. Finally, all existing multiple keywords must be unpacked using 256 as a base. These must then be repacked (maximum of two per word) using 512 as a base. This system allows the code to be executed on a computer with as little as 32 bits per word.

It is recommended that before attempts are made to include or modify tasks, the user become familiar with the input language processing by turning on the debug option with the ILP field specified. This will result in a detailed printout of exactly what takes place during the input language processing, the construction of the tables associated with the processing of the user's command stream, and the tables which are used during the task execution phase of the GEMACS code.

An example of the addition of a new command is presented here. Rather than reproduce the entire contents of the arrays to be modified, the additions to each array will be presented. The command, named COLPSE, to be added is to result in the elimination of the i^{th} row and column of a matrix. This would eliminate the i^{th} segment from a structure without regenerating the interaction matrix. Assume the form of the command would be:

SDS = COLPSE (DS) , ITH = n

where SDS is the resultant data set, DS is the previously defined symbol which contains the original data, and n is the index to the row and column to be eliminated.

The first step is to construct the NCODES array entries for COLPSE and ITH. This is done as previously described and is explicitly shown below.

$C = 3$
$CO = 3 * 2^6 + 15 = 207$
$COL = 207 * 2^6 + 12 = 13260$
$COLP = 13260 * 2^6 + 16 = 848656$
$COLPS = 848656 * 2^6 + 19 = 54314003$
$COLPSE = 54314003 * 2^6 + 5 = 3476096197$

$I = 9$
$IT = 9 * 2^6 + 20 = 596$
$ITH = 596 * 2^6 + 8 = 38152$

The values for COLPSE and ITH would then be added to the NCODES array in the first available positions. Therefore, NCODES (247) = 3476096197 and NCODES (248) = 38152.

The next step involves loading the keyword indices in the KEYWORD INDICES section of the PARTAB common block. This involves defining the variables KWCLPS and KWITH which will have the integer values of the KWNAME, KWARG, and KWFMTP array indices containing the keyword data. Assuming that the new data will be loaded at the end of the existing list, then COLPSE can use location 147 and ITH can use 148. Therefore, load in BLKDAT in the KEYWORD INDICES section:

```
DATA KWCLPS, KWITH/147, 148/
```

The corresponding locations in the KWNAME array then contain data which point to the location in the NCODES array containing these keywords:

KWNAME (147) = 247 (Pointer to the NCODES array)
KWNAME (148) = 248 (Pointer to the NCODES array)

The corresponding locations in the keyword argument array contain data indicating the argument type associated with these keywords.

KWARG (147) = -2 (Defined symbol for argument of COLPSE)
KWARG (148) = -3 (Literal argument for ITH)

The corresponding locations in the format table pointer array would indicate the starting location of a task format if a task is associated with a keyword.

KWFMTP (147) = 111 (Pointer to NTSFMT array)
KWFMTP (148) = 0 (No task for ITH)

At this point, the variable KWMAX must be updated to 148 from 146 since the number of valid keywords is now 148.

Data must also be loaded into the NAMTSK array in subroutine BLKDAT. Their positions in the array need not correspond to the position of related data in KWNAME, KWARG, and KWFMTP. The contents of the array point to the position of the task name in the NCODES array. Since the position is arbitrary, it may be placed after the last existing entry. Assume that this entry is in position 47. Then,

NAMTSK (48) = 247

Finally, the entry starting in position 111 of NTSFMT must be made. Writing the command using the argument type codes listed in BLKDAT, we have:

```
SDS = COLPSE (DS)    ,ITH = n
-1,          -2       148
```

Since the pointer to the name of this task in the NCODES array is stored in position 48 of the NAMTSK array, positions 111 through 115 of the NTSFMT array would be loaded with:

NTSFMT (111) = 4 (Number of entries for task)

NTSFMT (112) = 48 (Task number)

NTSFMT (113) = -1 (Symbol as first argument)

NTSFMT (114) = -2 (Defined symbol as second argument)

NTSFMT (115) = 148 (Keyword ITH as third argument)

Based on these data added to the code, when the COLPSE command is encountered, GEMACS will perform the following functions. The parse function will load the task number (48) into the next position of the NARGTB array. This number will be used in a computed GO TO statement in TSKXQT to call the code to perform this function. Task numbers must be unique.

The next action will be to search for the DS argument for the COLPSE keyword which defined the task. Starting with COLPSE, subroutine FNDARG will search for the next previously defined symbol DS in the user input stream. If one is not found, a NOPCOD is loaded in array NARGTB and the search for SDS will begin; and if found, the index to the NDATBL array will be loaded into NARGTB. Again, if not found, a NOPCOD will be entered into NARGTB. The next argument to be retrieved is the keyword ITH. Since it is a keyword, the argument type will be retrieved from KWARG (148), and FNDARG will search for a literal following the ITH field. When found, the negative of the index to the literal table array LITNUM will be entered into the next NARGTB location.

Note that the absence of arguments does not cause an error in the Input Language Processor. It is the function of the code which executes the command to verify the adequacy of the data on pass 1 of the execution.

Assume that this command is the third command and that it starts at location 17 of NARGTB. Then NTSKTB (3) = 17 and NTSKTB (4) = 21 since this command has four arguments. The contents of array NARGTB will be:

NARGTB (17) = 48 (Task number)

NARGTB (18) = 5 (Indicating the location of DS in NDATBL)

NARGTB (19) = 8 (Indicating the location of SDS in NDATBL)

NARGTB (20) = -5 (Indicating the location of n in LITNUM)

The contents of the INTARG array when the subroutine is entered to execute the task will be:

INTARG (1) = 5

INTARG (2) = 8

INTARG (3) = -5

If items are omitted, the value associated with the variable NOPCOD will be present in the INTARG array. Additional ARGCOM entries would be NUMARG = 3, NUMTSK = 48.

The user must verify the correctness of the data in the subroutine added to perform the function. In addition, the user must modify TSKXQT updating the computed GO TO and inserting the code which will call the subroutine which performs the function.

D. GEMACS TASK EXECUTION PROCESSORS

After completion of the input language process or module initialization, the control is returned to the executive routines. If no errors have occurred, the task execution processor subroutine TSKXQT is called to execute those commands present in the task list stored in the NARGTB array. Execution takes place in two passes. During pass 1 the arguments present in the NARGTB array are checked to make sure that they are consistent, that the required arguments are present, and that no table overflows will occur during execution. The pointer to the first argument for each task in the NARGTB array is stored in the NTSKTB array. The next entry in the NTSKTB array points to the first argument for the subsequent task. Thus, the arguments for the current task are bounded by the current entry in the NTSKTB array and the next entry in the NTSKTB array.

Before the task is executed, a call is made to subroutine SYSCHK to determine if the time for a checkpoint is past. If no CHKPNT command has been encountered in the user's input stream, no checkpoint will be written.

After the checkpoint status has been determined, the entries in the NARGTB array are transferred to the INTARG array stored in common ARGCOM. The number of entries transferred is stored in the variable NUMARG which is located in common ARGCOM. Control is then transferred on a computed GO TO, which uses the task number as the index. Some commands will be executed directly in the TSKXQT routine; however, most commands will be executed by a separate command processing subroutine. When the subroutine is called to execute a given command, it will retrieve the arguments from the INTARG array and perform the necessary operations to assure the validity and completeness of the arguments provided.

The individual task processors are described in the remainder of this section. Not every processor appears in every module; reference to figures I-1 through I-4 will allow the reader to determine where each processor appears.

1. GEMACS Geometry Processor (INPUT Module)

The GEMACS geometry processor is, in effect, an input language processor for geometry data. These data are separated from the task commands by an END command. The reason for processing the geometry commands separately is that the geometry command format, although it is free-field, is of fixed order.

The execution of the geometry processor is caused by a GMDATA command encountered in the user's input stream. The functional flow is presented in figure I-9. Control is transferred to subroutine GEODRV which initializes the storage area and data blocks necessary to process the input geometry. If the data set name specified is the same as the previous geometry name for the last call to GEODRV, it is assumed that the user is expanding the geometry data set and not creating a new data set. Consequently, the new data will be concatenated to the old data and, in effect, the structure being represented will be expanded.

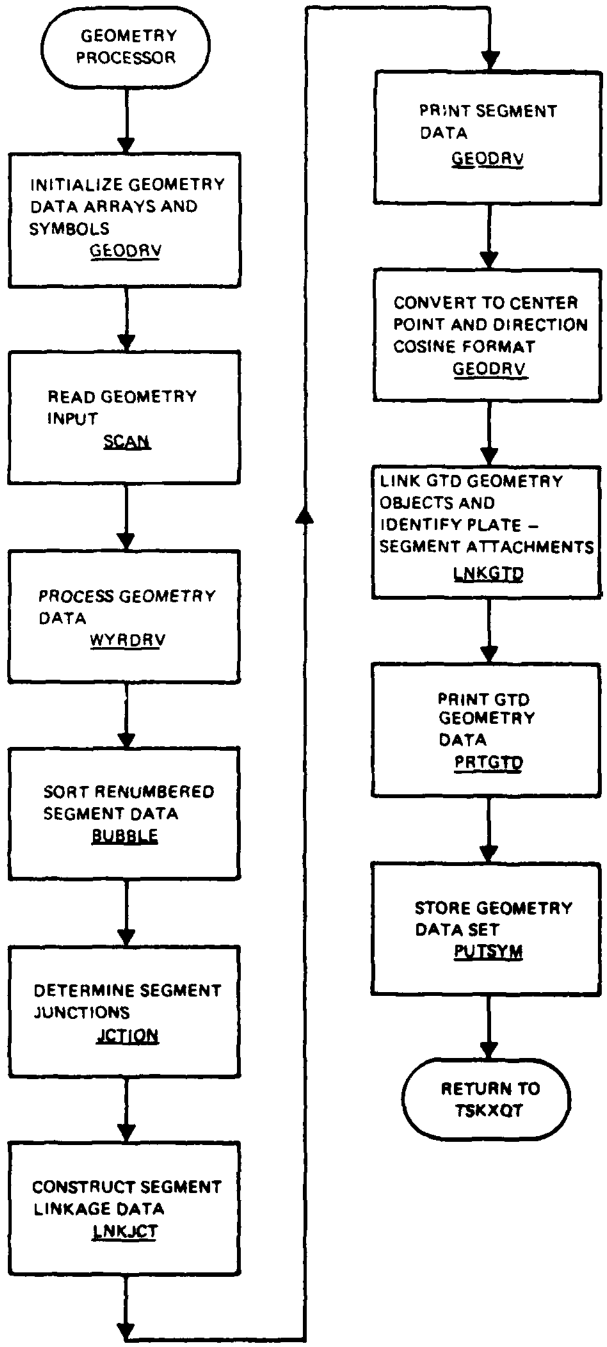


Figure I-9. Geometry Processor Function/Subroutine Interface

Subroutine GEODRV calls subroutine WYRDRV to process the geometry input. The subroutine WYRDRV performs basically the same function as the subroutines PARSE and FNDARG in the Input Language Processor. Subroutine WYRDRV uses subroutine SCAN to read the user's input. However, there is a basic difference in that common variable IGNORE is set to ISON for calls to SCAN from the geometry processor. This results in subroutine SCAN performing two operations differently for the geometry input. First, keywords will not be recognized if they are present in the input. That is, all alphanumeric input will be returned as alphanumeric, and secondly, monadic operators are not treated as separate fields. The sign of the subsequent numeric field following the monadic operator is changed to that of the monadic operator.

As stated previously, the inputs to the geometry processor are free-field; however, they must appear in a specified order for each command as discussed in the User Manual. Subroutine WYRDRV searches the NVAL array for the occurrence of one of the mnemonics associated with a geometry processor command. If such a mnemonic is found, control is transferred based on the mnemonic index to that portion of WYRDRV to process the geometry command. If the mnemonic entry is not found, a fatal error will occur, though processing will continue until an END command is encountered in the geometry input stream. In this way all errors in the user's geometry input will be detected during a single run.

After completing the reading of the user's input for the geometry processor, control is returned to subroutine GEODRV which calls subroutine BUBBLE to sort the SEGTBL array into ascending order. The segments specified during the geometry input must be in their corresponding location in the SEGTBL array; that is, segment 1 must be stored in the first location, segment 2 must be stored in the second location, and so forth. The SEGTBL array is sorted such that all wires are stored in ascending segment order, followed by all patches stored in ascending tag order. GTD elements fall to the end of the list. If there have been N segments or patches specified during the input geometry, subroutine BUBBLE must find all N segments or patches. Failure to do so will result in a fatal error.

After the SEGTBL array has been sorted, subroutine GEODRV will call subroutine JCTION to determine the junctions and patch connections for each wire segment specified. In subroutine JCTION the end points of each segment are compared with the end points of all higher numbered segments; if the end points are found to be within a specified tolerance of each other, they are assumed to form a multiple or single junction. This specified tolerance is the variable ZERO loaded in subroutine BLKDAT and should represert the round-off error limit for the computer on which GEMACS is being executed. If two wire segments are found to be connected to the same junctions on both ends, they are identical segments. In this case, the SEGTBL array entry for the higher numbered segment is set to 0. During subsequent execution of the geometry, all interactions with any segment number 0 will be set to 0 except for the self term which will be set to 1. Using this artifice of an augmented matrix, segments in a plane of reflection or on the axis of rotation are allowed; their excitation will be set to 0, resulting in a current of 0 on the segment. Thus the only segment which will be considered in the subsequent analysis will be the original segment on the axis of rotation or in the plane of symmetry.

Once all of the segment connections have been identified, subroutine LNKJCT is called to generate the circular linked list identifying the connnectivity of all segments. This is accomplished by taking the junction information for the first segment and searching all subsequent segments for identical junction information. When a segment is found with the same junction condition as the first segment, the junction word for the first segment is set to point to the segment number to which it is connected with a bias to indicate which end of the segment is connected to segment 1. The search is continued through all remaining segments to find the next segment connected at the junction for segment 1, and the junction word for the previously connected segment is then modified to point to the correct end of the next segment connected. When the last segment at the junction is found, its junction word is modified to point to the first segment at the given junction. This is accomplished for both ends of any given segment and results in a circular linked list such that the connectivity information for any segment points directly to the next segment connected at the junction. The connectivity information for the next segment points to the next segment connected, and this continues until the pointer links the last segment back to the first segment connected to a given junction.

Once all of the segments have been linked, control is returned to subroutine GEODRV which will then print the wire and patch data and convert from an end point representation of the segment and patch to a center point and direction cosine representation of the segment and patch. This final data form is then used throughout the remainder of the GEMACS code when geometry data are referenced.

The GTD geometry elements (plates, cylinders, and endcaps) processed by WYRDRV now appear at the end of the geometry data set. These elements must be linked together so that the GTD physics routines know which end cap is attached to which end of the cylinder and which (if any) segments connect to plates. Subroutine LNKGTD performs this function.

Control then returns to GEODRV, which calls subroutine PRTGTD to output a list of data on all GTD geometry elements. Finally, subroutine PUTSYM is called to store the geometry data on an external peripheral file.

2. Interaction Matrix Processor (GTD, MOM Modules)

The interaction matrix processor is interfaced with the task execution processor through subroutine ZIJDRV in the GTD and MOM modules. The flow for each module is illustrated in figures I-10 and I-11. In both modules this subroutine retrieves the arguments from the INTARG array in common /ARGCOM/. Default values may be used for the geometry data set name, the frequency, and the ground parameters. If the frequency has not been specified on the ZGEN command which causes execution of the subroutine ZIJDRV, then it must have been previously specified on another command or in a direct manipulation statement. In this case, the frequency is retrieved from the common storage area AMPZIJ. A value of 0 for the frequency will result in a fatal error. At this point the functions of ZIJDRV in the two modules differ and must be discussed separately.

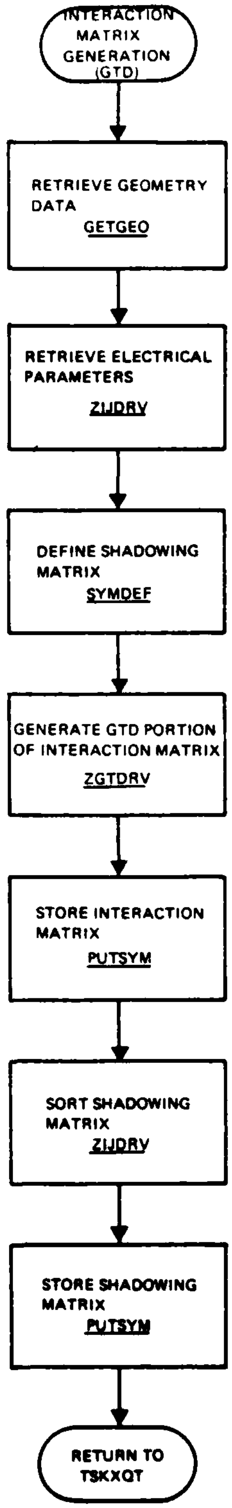


Figure I-10. GTD Module Interaction Matrix Generation Function/Subroutine Interface

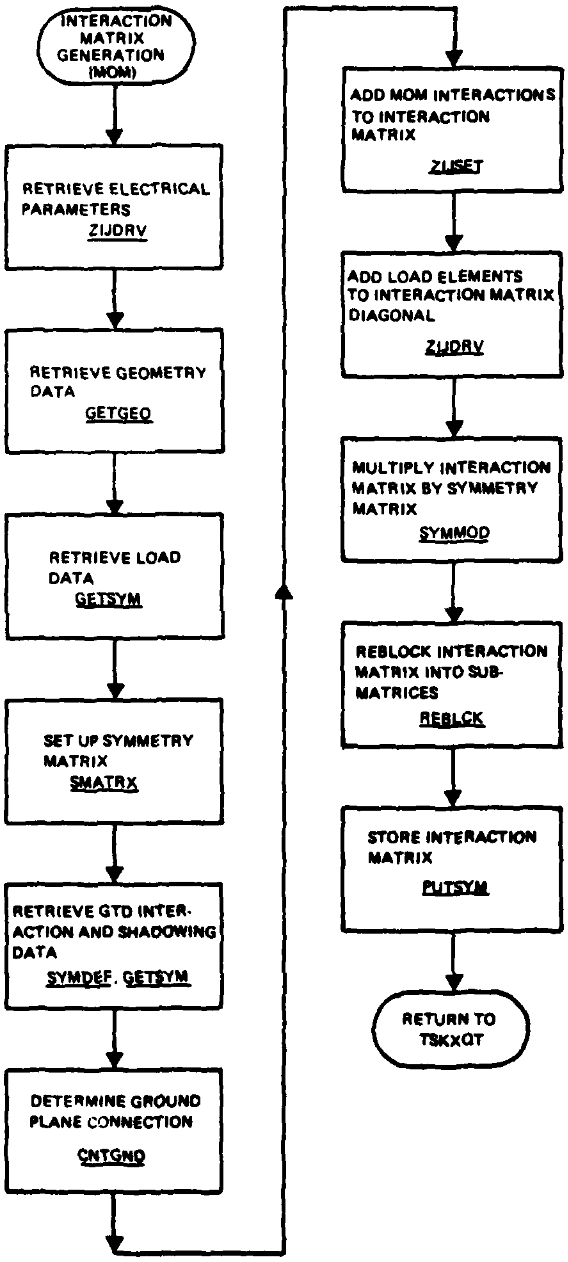


Figure I-11. MOM Module Interaction Matrix Generation Function/Subroutine Interface

a. <u>GTD Module Interaction Matrix Processing</u>

After the above preliminaries have been performed, the segment shadowing matrix is defined by SYMDEF. This data set will be filled later by ZGTDRV with integer pairs. The pairs define geometry segments (wires and patches) for which the direct path between a pair is obstructed by a GTD geometry object. The shadowing matrix is used by ZIJDRV and ZIJSET in the MOM module to indicate which MOM interactions should be set to zero due to this obstruction. One obstructed path corresponds to one entry in the shadowing matrix. The segment number of the source segment is placed in the left part of the entry, while the segment number of the observation segment is placed in the right part of the entry.

Next, ZGTDRV is called to generate the GTD portion of the interaction matrix. Subroutine ZGTDRV interfaces the GTD physics routines with the rest of the code. Since the entire interaction matrix will seldom fit in core at once, it is necessary to call ZGTDRV several times, once for each block of the interaction matrix which will fit in core. There must be storage available to generate at least two columns of the interaction matrix in core at one time. Since each element requires two words (complex data), the code requires 4N storage locations in the TEMP array, where N is the number of wires and patches for the geometry being modeled. If the structure is loaded, the MOM module will require an additional 2N locations, for an effective limit of 6N locations for a loaded structure.

After each block of the interaction matrix has been generated it will be written to its associated peripheral file using subroutine PUTSYM. The symbol in the NDATBL array will have its column attributes updated to reflect the total number of columns stored at this point. The process continues until the entire interaction matrix has been generated.

Note that the interaction matrix, as generated, is actually the transpose of the interaction matrix written down in a formal manner. The primary reason for this storage method is more rapid peripheral file I/O for matrix multiplication and other matrix operations.

b. MOM Module Interaction Matrix Processing

The MOM version of ZIJDRV also performs the data retrieval functions discussed above. Furthermore, if a load data set has been specified, the data associated with the structure loads are retrieved and stored in the first available cells of the TEMP array.

If symmetry can be taken advantage of, subroutine SMATRX is called to generate the symmetry operator matrix in the next available cells of TEMP. The rank of the symmetry matrix is equal to 2^M, where M is the number of symmetry reflections or rotations. More information on symmetry may be found in sections F.2 and F.3 of the Engineering Manual.

The next step is to retrieve the interaction matrix and shadowing data previously computed by the GTD module. If no GTD interactions have been selected, this step is bypassed. As the formats of the GTD and MOM matrices are identical (complex, transpose), the MOM data will be added to the GTD data. When no GTD interactions have been selected, the interaction matrix is zeroed initially instead.

Whenever a ground plane has been specified, it is necessary to identify which wire segments, if any, are connected to ground so that the method of images may be applied correctly to the pulse + sine + cosine basis function. Subroutine CNVAMP performs this function by modifying the connection data in the SEGTBL array for any grounded segments.

After the frequency, ground parameters, geometry data, and load data have been retrieved, the interaction matrix is defined and the attributes determined. For the initial definition of the interaction matrix, the number of columns is set to zero. This is in order to enable a checkpoint/restart to be executed out of subroutine ZIJDRV. After the symbol associated with the interaction matrix has been defined, the common area in array TEMP available to store the elements of the interaction matrix is determined. There must be storage available to generate at least two columns of the interaction matrix in core at one time. (This is in addition to the column of storage required in the TEMP array for the load data.) Since each element requires two words due to the fact that the data are complex, the code requires 6N storage locations in common array TEMP for a loaded structure, and only 4N locations in the TEMP array for an unloaded structure, where N is the number of wires and patches for the geometry being modeled.

After the available storage has been determined, the number of columns which may be generated at any one time in core is determined, and subroutine ZIJSET is called to generate each block of the interaction matrix. Upon return from ZIJSET, the loads specified for the structure are added to the diagonal elements of the interaction matrix. If symmetry can be utilized, the interaction matrix is multiplied by the symmetry matrix (subroutine SYMMOD) and reblocked into square submatrices (subroutine REBLCK). This format allows rapid solution of the matrix problem by the solution processor.

After the current block of the interaction matrix has been generated with its associated loads, the symbol in the NDATBL array associated with the interaction matrix data will have its column attribute updated to reflect the total number of columns of the interaction matrix which have been generated at this point. Once this is accomplished, the current block of the interaction matrix will be written out to a peripheral file. This procedure will continue until all columns of the interaction matrix have been generated.

c. Communication Between GTD and MOM Versions of ZIJDRV

The division of the interaction matrix generation function between two GEMACS modules makes necessary a method of transferring data generated in the GTD module to the MOM module. This is done via the checkpoint file, as discussed under module initialization (secton I.B.1). In this way data generated by ZIJDRV (GTD) are available to ZIJDRV (MOM).

The easiest example to understand is the case in which all capabilities of the code are selected, both MOM and GTD interactions and both MOM and GTD objects in the geometry. In this case ZIJDRV (GTD) generates the interaction matrix for GTD interactions. The matrix is stored on the peripheral file associated with that symbol until module termination, at which time that peripheral file is written to the checkpoint file. The MOM module is initialized from that checkpoint file, and the GTD-generated data are placed back onto a peripheral data file. It is this data set which is retrieved into core by ZIJDRV (MOM). The MOM module adds the unobstructed method of moments interactions to the (previously generated) GTD interactions, and thus the complete interaction matrix is generated.

Two special cases can occur, and these will modify the flow of the ZIJDRV subroutines slightly. The first case occurs when the user requests GTD interactions, but no GTD objects are in the geometry. In this case a zero GTD interaction matrix is generated and passed to the MOM module via the checkpoint file.

The second case occurs when the GTD module is executed, but no GTD interactions have been selected. This time no data are generated and no peripheral file is used because the exercise of the GTD module should be transparent if no GTD interactions have been selected. (If the GTD module were omitted from the execution stream, no data would be generated either.) In this case, the MOM module defines the interaction matrix data set and initializes it to zero itself.

3. Matrix Solution Processor (MOM Module)

The solution of the set of simultaneous linear equations which results from the reduction of the electric or magnetic field integral equations using the method of moments formalism is carried out in three primary routines in the GEMACS code. These routines will perform the lower/upper triangular decomposition on the interaction matrix (LUDDRV), the forward elimination and back substitution on the decomposed interaction matrix (DECOMP), and the iteration until convergence or divergence is indicated (SOLDRV).

The decomposition of the matrix is interfaced through subroutine LUDDRV which will retrieve the information and the data for the subroutine DECOMP. This task is illustrated in figure I-12. Upon entry, subroutine LUDDRV will generate two auxiliary symbols to be associated with the symbol name of the matrix to be decomposed. These symbols will be formed by removing the rightmost three characters of the symbol name and appending the characters LWR and UPR which will represent the lower

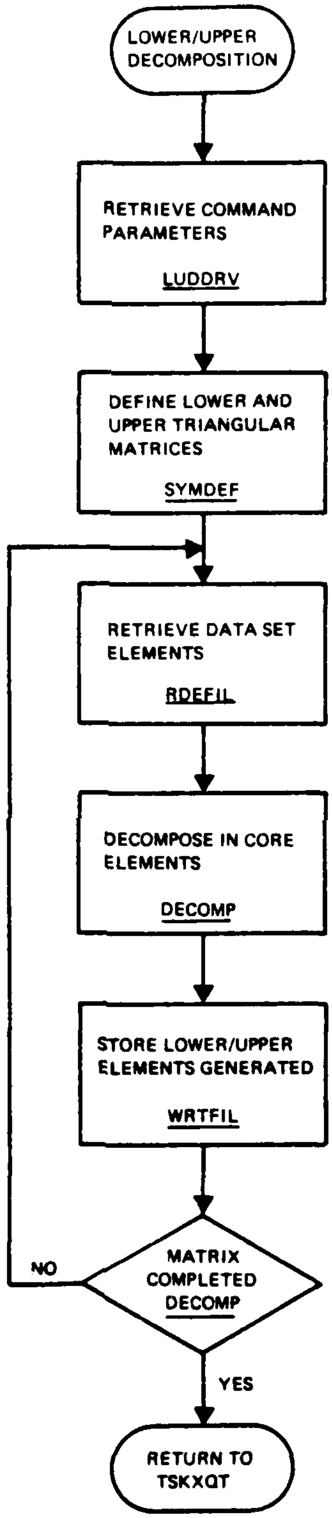


Figure I-12. Lower/Upper Decomposition Function/Subroutine Interface and upper triangular decomposed matrices. These symbolic names may be addressed in PRINT and WRITE commands in the command language. They may also be referenced in a PURGE command in order to make the peripheral files, which they occupy, available for other data when the lower and upper triangular matrices are no longer needed. LUDDRV also temporarily destroys the data in the array SEGTBL. The area formerly occupied by these data is now used to perform the decomposition operations.

All matrices are decomposed using the same subroutine. Subroutine DECOMP will decompose a banded or nonbanded, real or complex, in core or out-of-core matrix. DECOMP is one GEMACS subroutine which interfaces directly with subroutines RDEFIL and WRTFIL without passing through subroutines GETSYM and PUTSYM. This is in order to increase the efficiency of the code by cutting down on extraneous and unnecessary computations which would be done in subroutines GETSYM and PUTSYM since they would be called alternately for each row and column of the lower and upper triangular matrix as the matrices are generated via the decomposition of the source matrix. Subroutine DECOMP must be provided storage for at least three columns of the original matrix to be decomposed. In general, this presents no problem since there must have been this much storage available in order to generate the interaction matrix. However, if subroutine DECOMP is being used to decompose a matrix which was generated exterior to GEMACS, the storage considerations must be recognized by the user.

During the decomposition of the matrix, subroutine DECOMP will reference FORTRAN logical units 1 and 2, identified internally as IOS1 and IOS2. Once it has been determined that the entire matrix remaining to be decomposed will fit in core, the I/O of the component parts of the lower and upper triangular matrix is halted until the entire matrix has been decomposed. Until this occurs, the rows of the lower and upper triangular matrices are written to their peripheral files as they are generated. The elements of these rows and columns are the elements in the pivot row and column of the matrix. Diagonal pivoting is used during the decomposition. The remainder of the matrix beneath the diagonal is all that will be required for the subsequent decomposition about the next diagonal element; therefore, these elements are written to the peripheral file which is not being used as the source of the matrix for the current round of decomposition.

When the current round of decompositon is completed, the peripheral files IOS1 and IOS2 are interchanged, and the output file for the previous decomposition becomes the input file for the current decomposition, and the input file for the previous decomposition becomes the output file for the current decomposition. These scratch files are flip-flopped in this way until the entire remaining matrix resides totally in core, at which time there is no need for further I/O to the peripheral devices until the decomposition is completed. When completed, any remaining elements of the upper triangular matrix are written to the peripheral file associated with it, and any remaining elements of the lower triangular matrix are written to its peripheral file.

Checkpointing may occur during the decomposition of a matrix. It is for this reason that the symbol name entries in NDATBL for the symbols associated with both the lower and upper triangular matrices are updated to reflect the actual number of rows and columns generated by the decomposition each time a row or column is written to its respective peripheral file.

Once the matrix has been decomposed into its lower and upper triangular components, the solution is obtained by forward elimination and back substitution. This process, illustrated in figure I-13, is controlled by subroutine SOLDRV which also controls the BMI solution process. The command data are retrieved by SOLDRV and the determination is made regarding the process involved. If there are three arguments, a BACSUB command is being executed; otherwise a BMI solution is being sought. For both cases, the RHS is retrieved through GETSYM; and, if a BMI solution is being processed, the original matrix is retrieved and sent to BMIRHS to compute the term RHS = RHS - (L+U) I_0, where I_0 is the solution obtained from the last iteration. I_0 is initialized to zero unless specifically initialized by the user by the SET command. This is

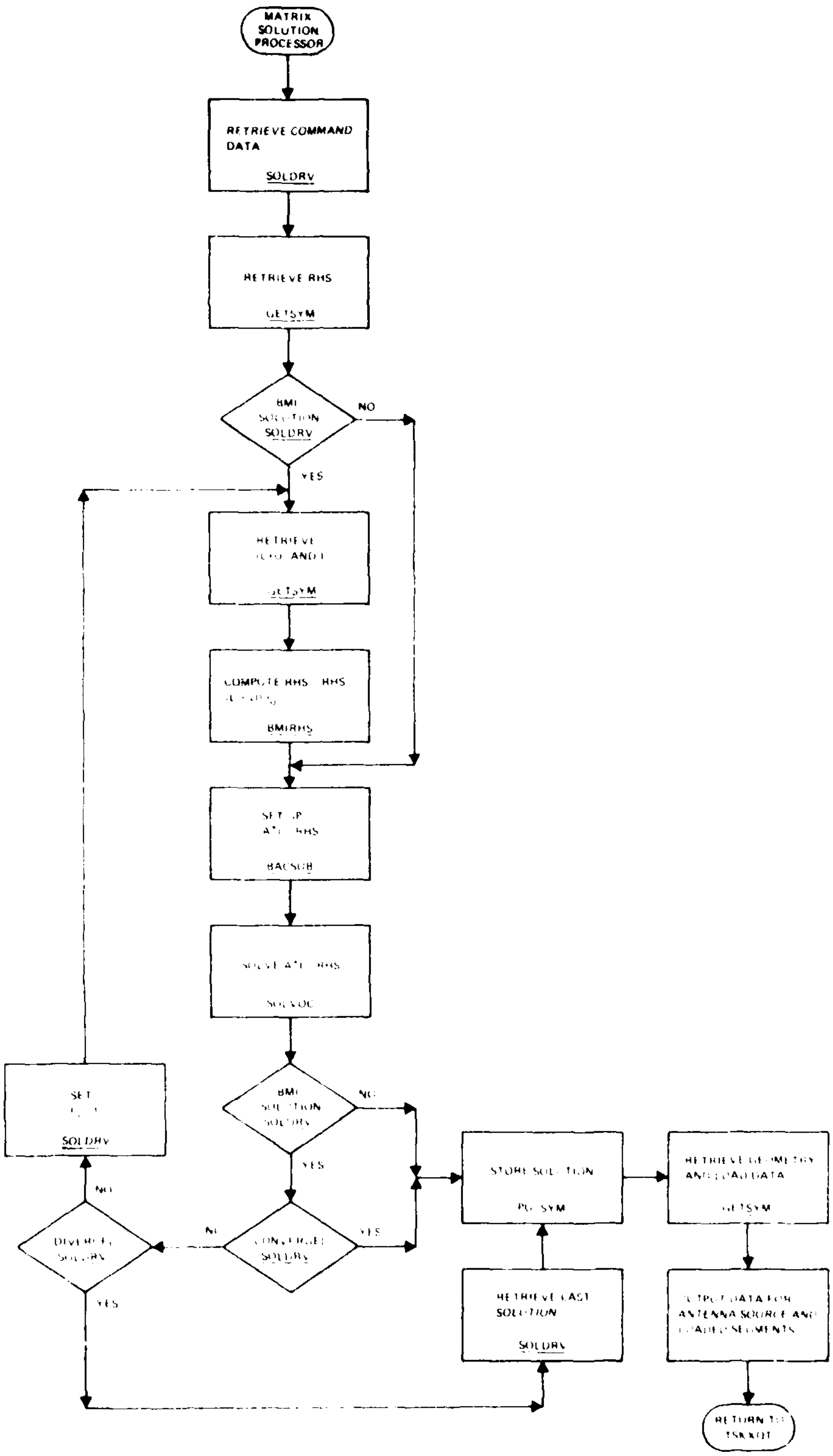


Figure I-13. Matrix Solution Processor Function/Subroutine Interface accomplished by specifying a different data set name for the solution on the left-hand side of the BMI command than that used for multiplying the (L+U) term on the right-hand side.

For both cases, the solution for the equation A * I = RHS is obtained through BACSUB, which will buffer in the components of A which fit in core and call SOLVOC to determine I. The actual data for I will overwrite RHS. At this point, if BACSUB is being executed, a jump is made to the output portion of the code. Otherwise, the convergence criterion is checked. If convergence has occurred, a jump is made to the output portion of the code. If convergence has not occurred and divergence is not indicated, a new RHS is computed as RHS = RHS - (L+U) I and the entire process is repeated. If divergence is indicated, the previous solution is retained and a jump to the output portion of the code is made.

The output portion stores the final solution and recovers the geometry and load data associated with the initial RHS. The electrical parameters for antenna source segments and loaded segments are computed and output to the user. Control then returns to TSKXQT.

4. Excitation Processors (GTD, MOM modules)

a. MOM Module Excitation Processing

The structure excitation processor, illustrated in figure I-14 allows the user to specify voltage, and/or plane and spherical wave excitations. Any superpositioning of these three sources may be utilized. Voltage excitation is converted to the E-field representation using a delta-gap source model. The electric field to be used as the source is simply the voltage applied to the source segment divided by the segment length. The implications of this type of model are discussed in section C.2.a of the GEMACS Engineering Manual. If a segment is excited using a voltage source, it is identified as an antenna driven segment by a 1 in row 11 of the SEGTBL array for that segment. The E-field representation is then derived by dividing the user-specified input voltage by the segment length stored in row 10 of SEGTBL.

The wave type of excitation may be specified with or without a parameter indicating the radial distance to the source. If the

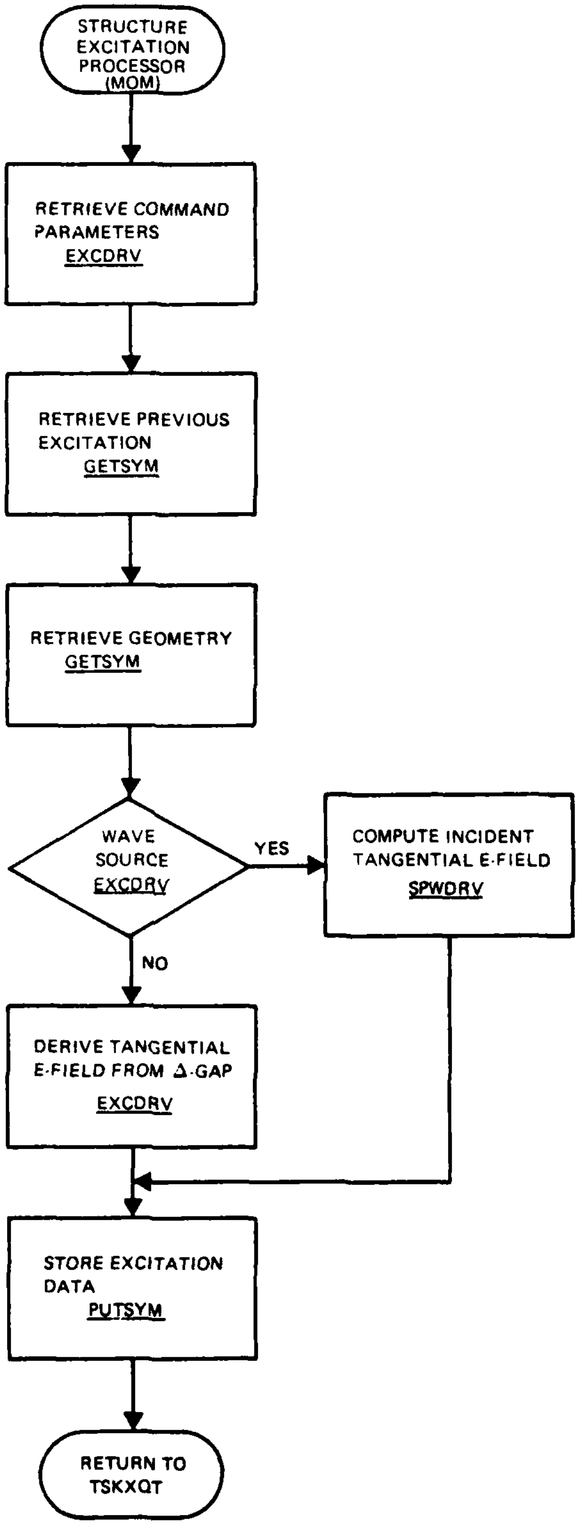


Figure I-14. MOM Module Excitation Processor Function/ Subroutine Interface radial parameter is not specified, a plane wave excitation will be assumed. If the radial parameter is specified, a spherical source will be located at the spherical coordinate specified on the EFIELD command. Both types of wave excitation may be used in the presence of a ground plane. The reflected waves will be computed using the modified Fresnel reflection coefficient as described in section C.2.b of the Engineering Manual.

The excitation source is represented as the vector sum $E = \bar{E}_1 \pm j\epsilon\ \bar{E}_2$. The magnitude of $|\bar{E}_1|/|\bar{E}_2|$ is the eccentricity ϵ. The polarization is indicated by the algebraic sign of the eccentricity as specified on the user's input. The direction of the propagation wave vector, $\bar{k}$, is given by the vector product $\bar{E}_1 \times \bar{E}_2$. For plane wave excitation, $\bar{k}$ is constant; and for spherical wave excitation $\bar{k}$ will always be oriented toward the midpoint of the wire segment or patch surface being excited.

The total excitation vector will be accumulated for each frequency and geometry data set specified. When either the frequency or geometry is not the same as that specified on the last entry into subroutine EXCDRV, the excitation vector will be reinitialized to zero.

Upon entry, the subroutine EXCDRV retrieves the command parameters (from the INTARG array), the previous excitation (if the frequency has not changed), and the geometry data set. If the frequency has changed, then all previous excitation data are zeroed out. Any portion of the excitation previously computed in the GTD module will be added to the previous excitation (or zero) data.

If the excitation is a spherical or plane wave, the subroutine SPWDRV will compute the incident electric field at the structure. It will also compute the ground reflected component if a ground is specified and add this to the direct wave at the structure.

If the excitation is a voltage source for an antenna, then EXCDRV will compute the tangential electric field on the segment using the delta-gap model. The field is derived as the voltage specified by the user divided by the length of the segment.

For either type of excitation, the data are stored in array TEMP. If both types of excitation are present, then the summing of sources for any segment is performed in the transfer of the data to TEMP. Finally, the data are transferred to the symbol specified by the user.

Note that if a structure excitation has been previously computed by the GTD module, the MOM module computation of the same excitation is suppressed.

b. GTD Module Excitation Processing

The GTD module excitation processor computes the tangential electric and magnetic fields on the structure being modeled due to wave-type sources in the presence of GTD geometry objects. The processor is illustrated in figure I-15. The operation of the GTD module processor follows that of the MOM module processor quite closely. Only significant differences will be discussed here.

The GTD module version of the excitation processor first examines the contents of the geometry being excited. If there are no MOM objects in the geometry, an excitation vector cannot be generated. EXCDRV defines an empty data set in this case so that any subsequent EFIELD command can use the data set name as input to compute incident fields.

Next the interaction array is checked to see if any GTD interactions were specified. If not, or if there are no GTD objects in the geometry data set, the GTD contributions to the excitation are zero. In this case, the MOM module performs the excitation generation function.

The excitation from a wave source is computed by a call to ZGTDRV. EXCDRV sets up the arguments to be passed to ZGTDRV, and ZGTDRV returns the excitation to be added to the TEMP array. Voltage excitation is performed only by the MOM module excitation processor.

c. Communication between GTD and MOM Versions of EXCDRV

Unlike the interaction matrix processor previously discussed, there is no direct communication between the two modules for excitation processing. Wave-type excitations are performed either completely in the GTD module or completely in the MOM module, depending on

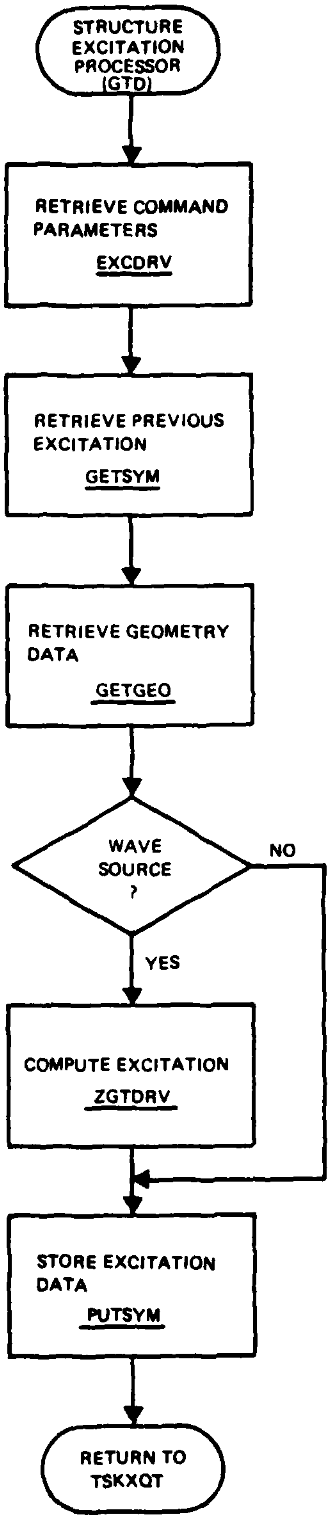


Figure I-15. GTD Module Excitation Processor Function/ Subroutine Interface the presence or absence of GTD objects or interactions. Voltage-type excitation is performed only by the MOM module.

5. Field Pattern Processors (GTD, MOM, OUTPUT Modules)

The field pattern processors are used to compute the near- and far-field responses of the structure being excited, with or without the addition of the fields incident from wave-type sources. When no GTD or incident field interactions have been requested, the MOM and OUTPUT modules perform this function. If either or both GTD or incident field effects are desired, the GTD module is also required.

a. GTD Module Field Pattern Processing

The field pattern processor for the GTD module is illustrated in figure I-16. Subroutine FLDDRV examines the INTARG array and retrieves observation points and geometry data set attributes.

If GTD interactions are specified and the second data set in INTARG is a solution data set, the Green's function matrix is computed by ZGTDRV and stored by PUTSYM. The symbolic name for the Green's function matrix is XXXGFM, where the Xs represent the first three characters of the field matrix symbolic name.

If incident fields were specified or if the second INTARG data set is a source data set, the incident field matrix is computed by ZGTDRV. Otherwise the incident field matrix is set to zero.

For GTD-only problems, the incident field data set may be passed directly to the OUTPUT module. The Green's function matrix is not generated. For MOM/GTD hybrid problems, the MOM module takes the Green's function matrix, multiplies it by the solution currents and adds it to the incident field to obtain the total field.

b. MOM Module Field Pattern Processor

The field pattern processor for the MOM module is illustrated in figure I-17. If GTD or incident field interactions have been specified, control is passed to subroutine EGFMAT to generate the total field pattern from the Green's function matrix, solution vector, and incident field vector. Otherwise, subroutine FLDDRV retrieves the solution (currents) and geometry data and then calls subroutines CABC,

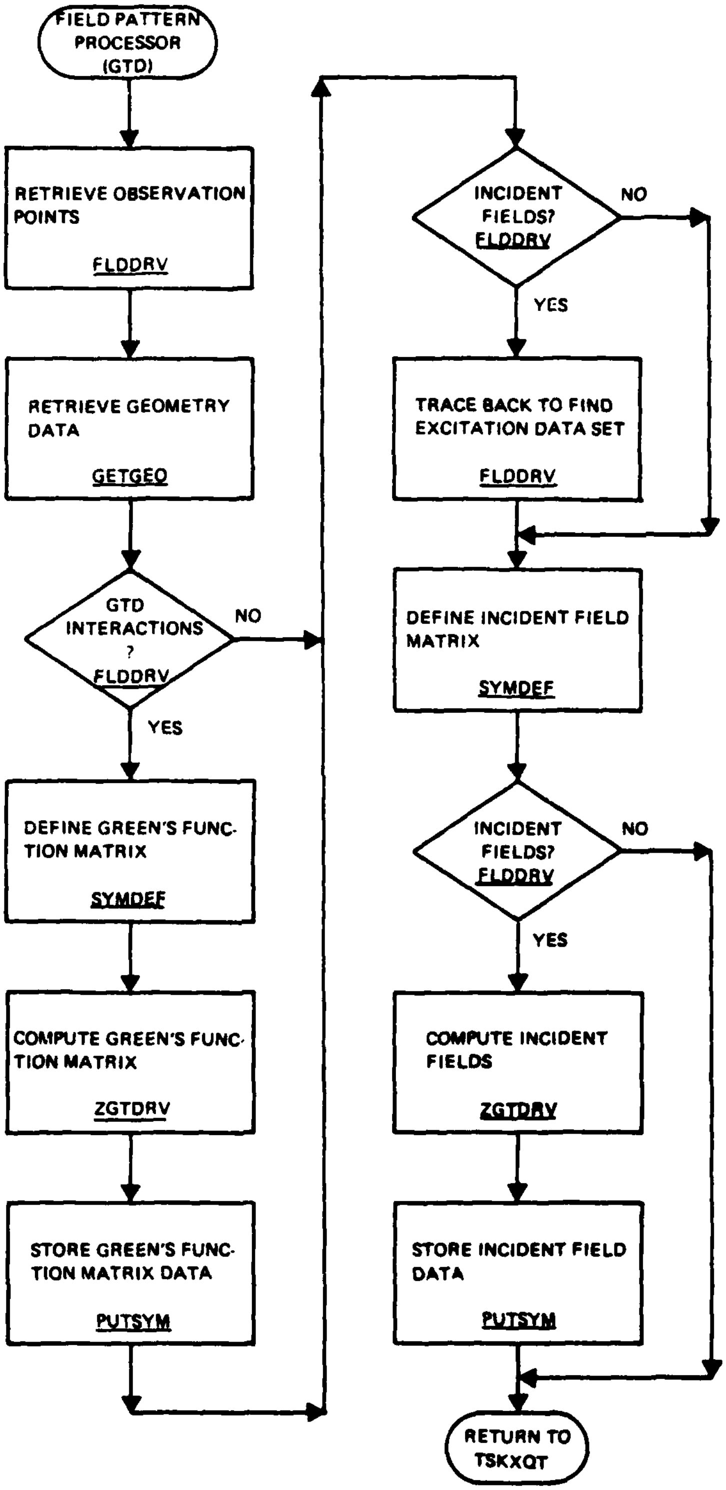


Figure I-16. GTD Module Field Pattern Processor Function/ Subroutine Interface

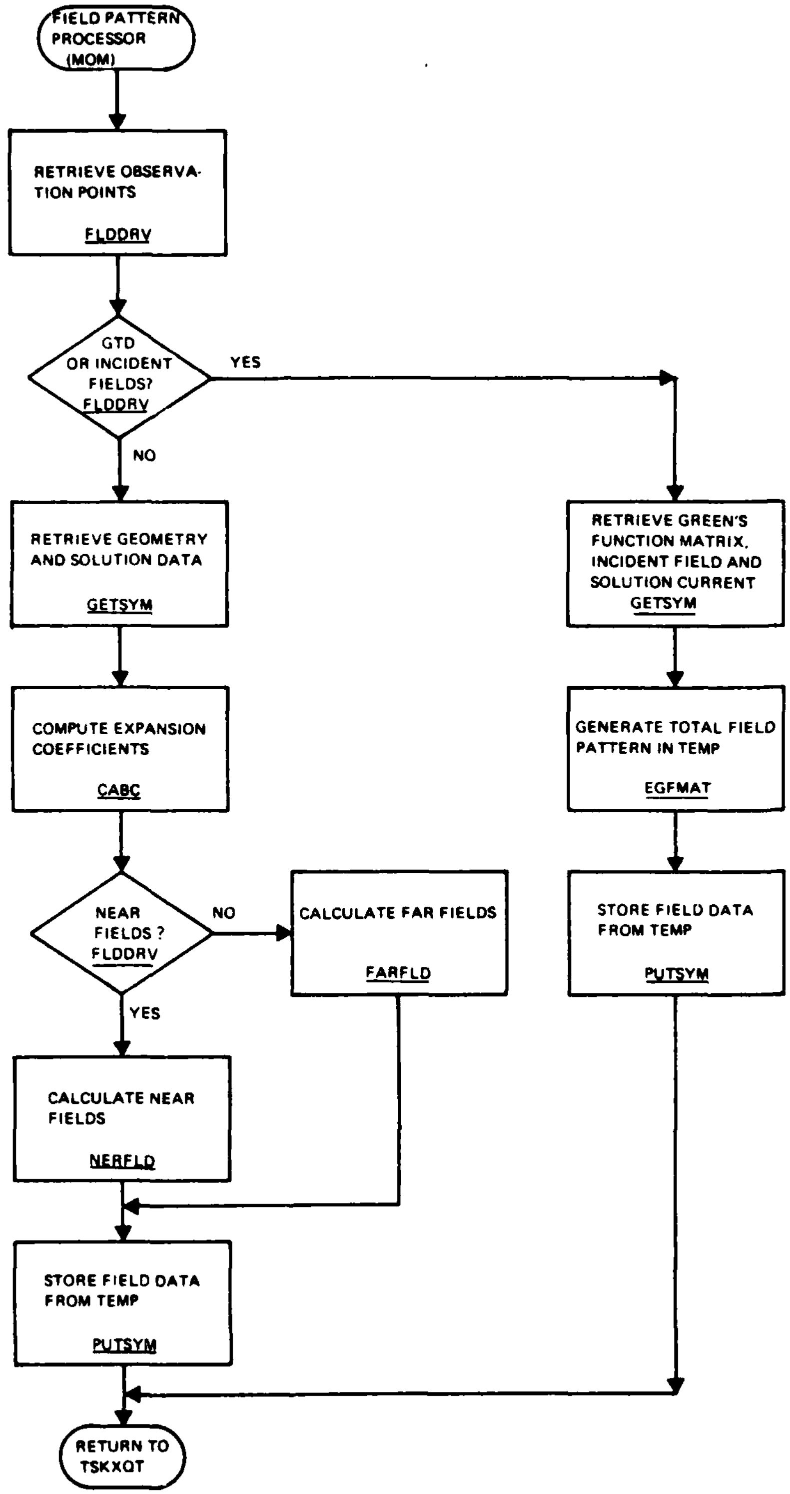


Figure I-17. MOM Module Field Pattern Processor Function/ Subroutine Interface

NERFLD, and FARFLD to compute the expansion coefficient of the current, the near electric field component, and the far electric field component, respectively. These subroutines function exactly as they did in their host program, AMP. After the field components have been determined, they are translated to the coordinate system specified by the user and stored on the field data set peripheral file in real and imaginary form.

c. OUTPUT Module Field Pattern Processor

The field pattern processor for the OUTPUT module is shown in figure I-18. After observation points have been determined by FLDDRV, the real/imaginary field data set generated by either the GTD or MOM module field pattern processor is retrieved, converted to magnitude/phase format, and restored. Control is then transferred to FLDOUT which will list the geometrical and field parameters and, if requested, plot the data in the format specified by the user.

d. Communications Among Field Pattern Processors

Figure I-19 shows how data are passed from one module to the next in order to generate field data. Remember that this figure shows only the concept of data flow. Actual data transfer takes place via the checkpoint file MODCHK.

For a GTD-only problem, the GTD module generates the incident field matrix which may then be passed to the OUTPUT module for plotting and printing. The MOM module has no effect in this case and may be omitted if not needed for another portion of the problem.

For MOM-only problems, the generation of the Green's function matrix and incident field matrix is suppressed unless incident fields have been requested by the SETINT command. In the former case the GTD module has no effect and may be omitted. In the latter case the GTD module is required. In either case both MOM and OUTPUT modules must be present.

For MOM/GTD hybrid problems, all modules are required. In this case the MOM module takes the Green's function matrix and incident field matrix generated by the GTD module and, with the solution currents found in the MOM module prior to the EFIELD command, generates the field pattern matrix.

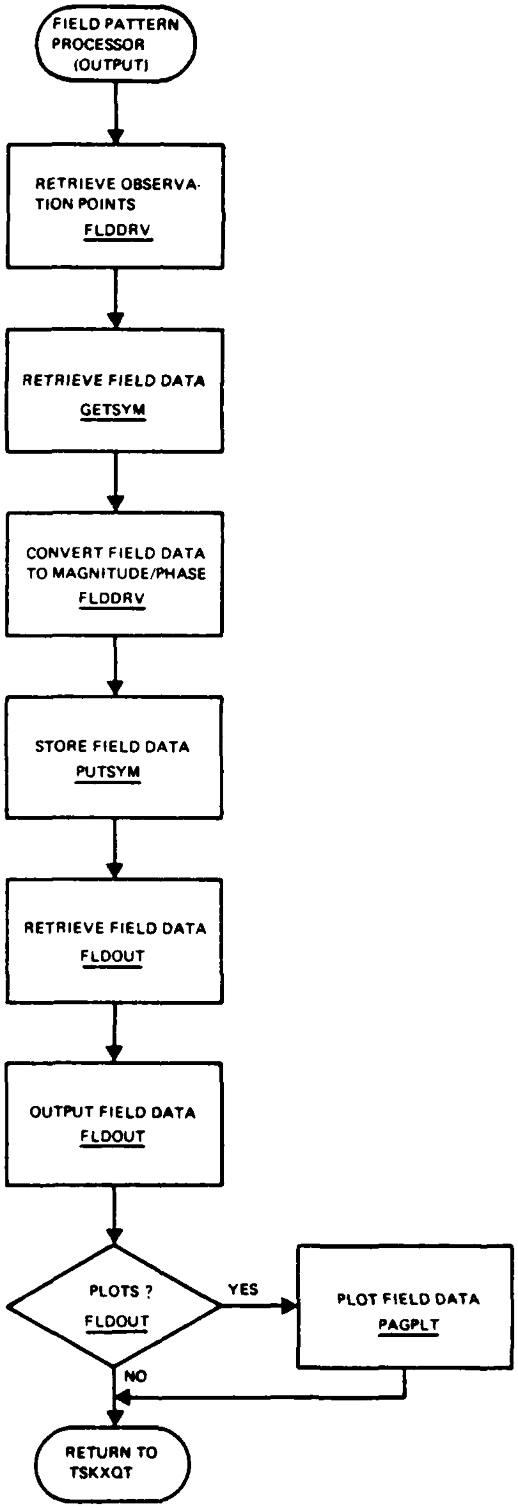


Figure I-18. OUTPUT Module Field Pattern Processor Function/ Subroutine Interface

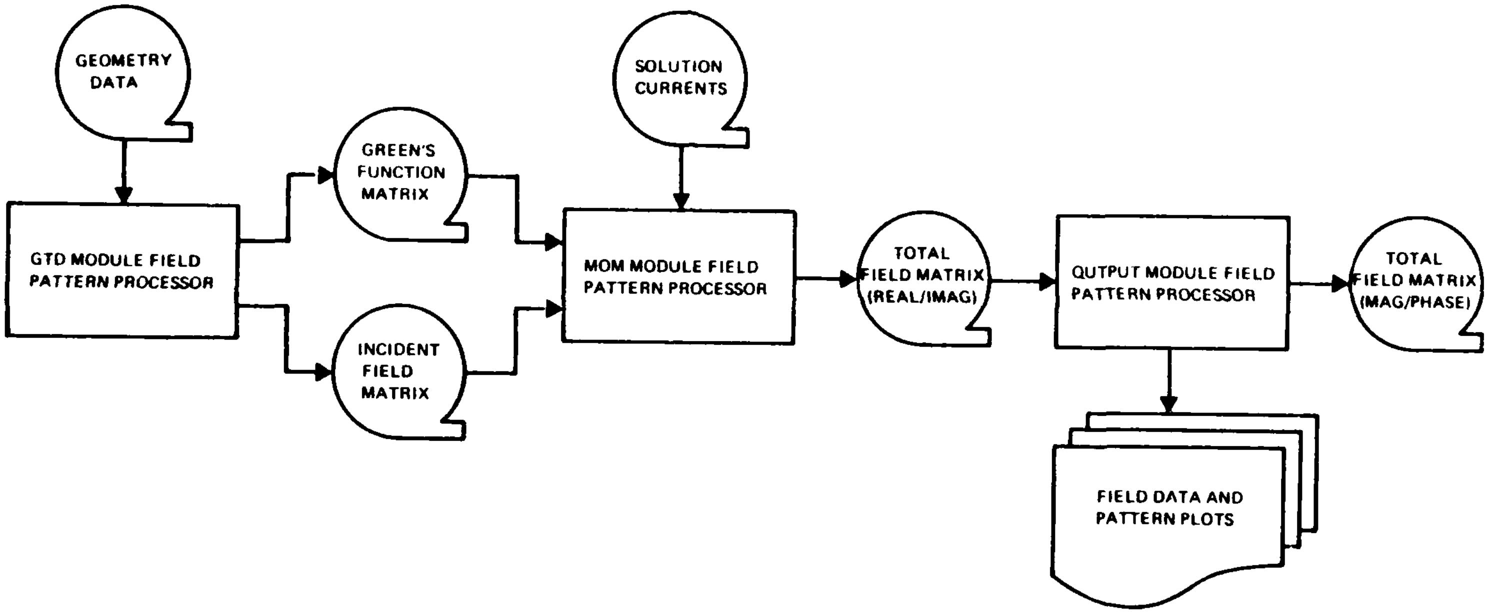


Figure I-19. Data Flow Through the Field Pattern Processing Module

6. Load Processor (MOM Module)

The generation of electrical loads on the structure is the function of the Load Processor illustrated in figure I-20. Subroutine LODDRV retrieves the command parameters specified by the user. The geometry data are retrieved; and, if the current frequency is the same as the previous frequency, the data, if any, for the previously generated loads are retrieved. If the frequency has changed since the last entry or there are no previously generated data, the load data set is initialized to zero. The values of the load impedance are computed and accumulated into the load data set. Upon completion, the load data set is stored and control is returned to TSKXQT.

7. Direct Manipulation Processor (All Modules)

The direct manipulation processor is used to process all command strings encountered in the user's input which do not contain a keyword relating to a specific GEMACS task. Thus, all commands which may have a misspelled keyword that would identify the task will be processed by the direct manipulation processor. This situation will typically be detected during execution because of illegal operand and operator fields present. Under normal conditions, the direct manipulation processor will scan the input from the user's command and determine the operations to be performed. The present capability includes only the mathematical operations of multiplication, division, addition, subtraction, and exponentiation. In addition, version 3 of GEMACS is limited to performing these operations on single dimension data. That is, no matrix operations are allowed. The code as constructed in the direct manipulation routine, subroutine DMPDRV, will recognize the need for matrix operation. The software to support the matrix operation has not been included.

The operations within subroutine DMPDRV are performed in a right to left search. There is no hierarchy of operation implemented. Therefore, operations which require exponentiation must have the exponent and the operand enclosed in parentheses. When a right parenthesis is found, the preceding operands and operator are recovered and the operation performed. The result of each operation is stored in an intermediate result until a left parenthesis is encountered. At this time,

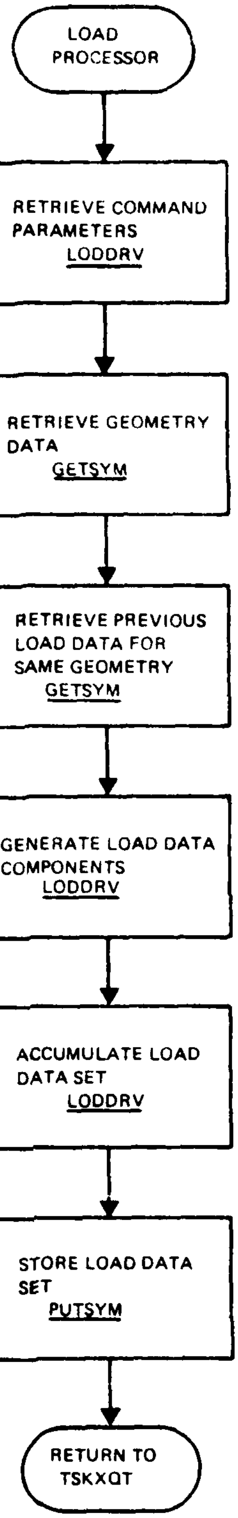


Figure I-20. Load Processor Function/Subroutine Interface the resultant data are identified as a symbol and entered into the symbol table. The next right parenthesis is then located, and the process is repeated. This continues until there are no more right parentheses, at which time the remainder of the string or input is processed until an equal sign is encountered. When an equal sign is encountered as the operator, the remaining data are stored under the symbol or keyword specified.

The primary function of the direct manipulation processor is to enable the user to load data into the internal variables associated with keywords such FRQ, COND, and EPSR. In addition, the keywords TIME and NUMFIL may be set in a direct manipulation statement. Using this processor, such items as the frequency or ground conductivity may be redefined during the execution of the code as algebraic functions of previous definitions of the same variable or as simple algebraic statements. The code is also intended to support the arithmetic matrix operations and the mixed scalar/matrix arithmetic operations. However, as stated previously, the code for this has not been implemented.

8. Interaction Processor (GTD, MOM, OUTPUT Modules)

The interaction processor is illustrated in figure I-21. Its purpose is to decode the keywords in the INTARG array in order to select which physics processes will be used in generating interaction, excitation, and field pattern data.

The heart of the process is the ISETTB array shown in table I-2. It consists of five columns of data for each keyword which could be present in the SETINT command:

(a) The number of the keyword in the KWNAME array

(b) The row number of the next keyword in ISETTB

(c) The row number in ISETTB of the next keyword to search for if the present keyword is found in INTARG

(d) An integer indicating the major group of physics interaction for this keyword:

1 = plate GTD

2 = cylinder GTD

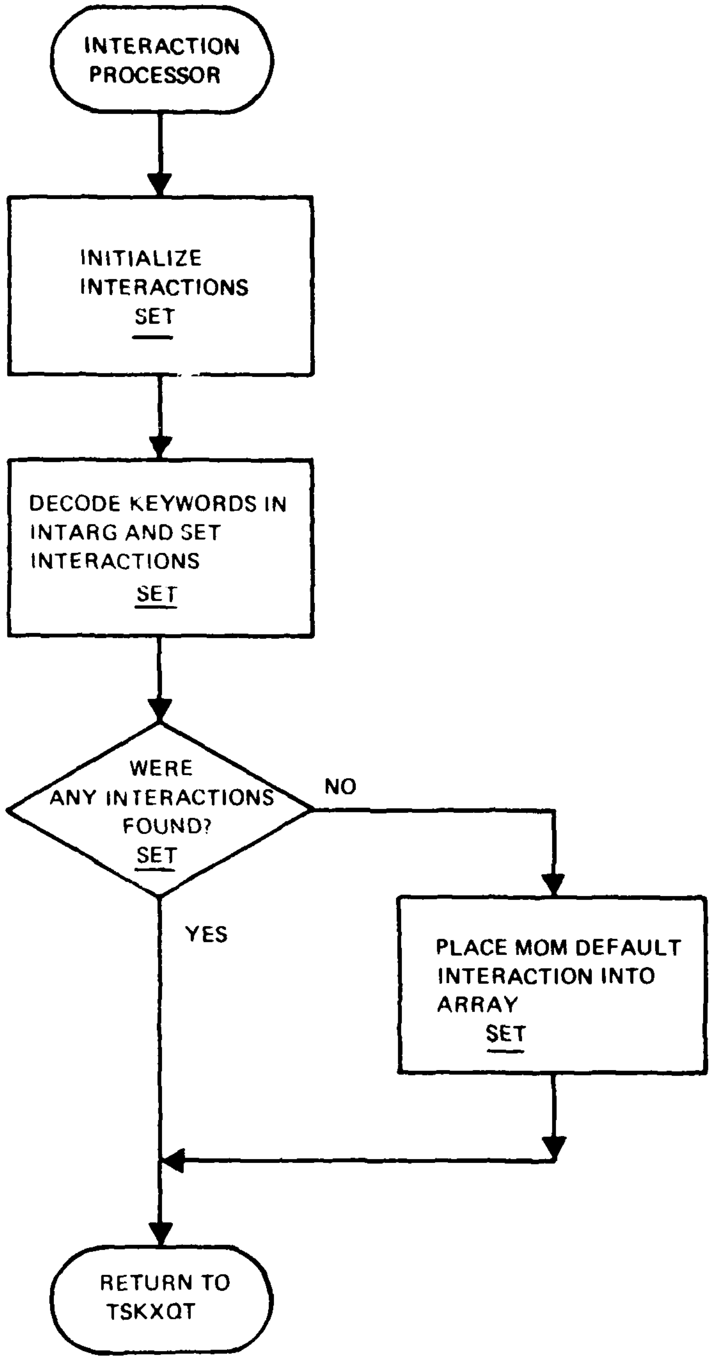


Figure I-21. Interaction Processor Function/Subroutine Interface

TABLE I-2. THE ISETTB ARRAY FOR SETINT COMMAND INTERACTION KEYWORDS

COLUMNS OF ISETTB

ROW (NUMBER)	1 (KEYWORD NUMBER)	2 (NEXT KW)	3 (STOP LIST)	4 (K)	5 (J)
1	136 (GTD)	2	18	0	0
2	127 (PL)	3	9	0	0
3	123 (PR)	4	4	1	2
4	126 (RR)	5	5	1	3
5	124 (PD)	6	6	1	4
6	125 (RD)	7	7	1	5
7	143 (PDR)	8	8	1	6
8	127 (PL)*	9	9	1	7
9	130 (CY)	10	13	0	0
10	141 (CS)	11	11	2	1
11	128 (ER)	12	12	2	2
12	129 (ED)	13	13	2	3
13	135 (PC)	14	18	0	0
14	131 (RC)	15	15	3	1
15	132 (CR)	16	16	3	2
16	133 (CD)	17	17	3	3
17	134 (DC)	18	18	3	4
18	137 (MM)	19	19	4	1
19	140 (EI)	20	22	0	0
20	138 (EU)	21	21	5	1
21	139 (ES)	22	22	5	2
22	0 --	0	0	0	0

*(K,J) = 1,7 sets the double diffraction flag but computes no new physics.

3 - plate cylinder GTD

4 = MOM

5 = Incident fields

(e) An integer indicating a particular type of interaction within a major group.

All interaction processing is performed by subroutine SET. All allowable SETINT keywords are stored in array ISETTB along with their (K,J) interaction indices. The argument list is examined for each of the keywords in ISETTB. When one is found, the corresponding values of K and J are packed into the next available word of the KJINT array. J occupies the first 16 bits of the entry. K occupies the next 16 bits. When the search is complete, a zero is loaded as the last entry of KJINT to terminate the interaction list.

Some keywords indicate more than one (K,J) interaction. PL, for example, covers the (K,J) cases of (1,2) (1,3) (1,4) (1,5) (1,6) and (1,7). When a keyword such as PL is encountered, all associated interactions are placed sequentially in KJINT. This is accomplished by use of the STOP LIST column of ISETTB. When a keyword is found that matches an ISETTB keyword, the (K,J) interaction indices of all keywords up to, but not including, the keyword pointed to by the STOP LIST index are placed in KJINT. (The (K,J) = (0,0) case is ignored.) For the PL keyword, the STOP LIST index is 9. Therefore, (K,J) interactions in ISETTB rows 2, 3, 4, 5, 6, 7 and 8 are placed in KJINT, and the keyword search continues for the keyword in row 9 (CY). This algorithm ensures that the KJINT list will always be in numerical order and that no duplication will occur.

The NEXT KW column of ISETTB makes it easy to add additional interactions to the SETINT command. For example, suppose the interaction of (K,J) = (1,8) were to be added with the keyword "R3" (triple reflection). The new keyword would be added in row 22:

22:	keyword # (R3)	5	5	1	8
23:	0	0	0	0	0

The next keyword entry in row 4 would become 22, as shown:

4:	126(RR)	22	22	1	3

The default interaction set is MM: (K,J) = (4,1) for method of moments interactions. This is initialized in BLOCK DATA for the INPUT module and by subroutine STRTUP for subsequent modules.

E. MODULE TERMINATION PROCESSOR

After completion of the tasks associated with the current module, GEMACS calls subroutine STATFN to output the accumulated timing information for the execution and to write the end of module checkpoint. The subroutine names listed are in order of decreasing execution time. The additional information printed out is the percent of the total accounted time for each subroutine and the number of times the subroutine was called. The total accounted time is printed out for the user's convenience. This time may be somewhat less than the actual run time noted in the user's day file. There are two reasons for the discrepancy: first, not all subroutines in the GEMACS code include the necessary coding to accumulate the timing information; and second, the timing algorithm on the host computer may be somewhat different for the run and accounting purposes. Those subroutines in GEMACS which have only one or two executable FORTRAN statements are not timed because the accumulation of the timing information would generally take longer than the execution of the subroutine itself.

In order to make use of the timing feature, the host computer must have a FORTRAN callable library subroutine to return the current CPU time. This library call occurs in GEMACS subroutine TICHEK. The library subroutine must return the argument T in seconds. If there is no library subroutine available to obtain the desired information, the code associated with subroutine TICHEK should be eliminated and replaced with a simple RETURN and END statement.

Note, if there is no library subroutine to return the current CPU time, the checkpoint feature can not be used with a time increment CPINC specified. In addition, the keyword TIME can not be set in the user's input since the current elapsed CPU time must be computable by GEMACS if this increment is specified.

The module termination processor also directs that a checkpoint be written to file MODCHK. This end-of-module checkpoint is identical to a timed or command checkpoint, and it is used in order to initialize the next module to be executed. In STATFN, the MODCHK file is rewound first to ensure that the end-of-module checkpoint is the only checkpoint on the file. This, of course, destroys any other checkpoints on the file. In most cases, a user would not want to retain them once the module terminated. Should the user wish to keep all intermediate checkpoints, he may do this by specifying an alternate file identifier on the CHKPNT command (IOCKPT≠MODCHK) and requesting the NR (no rewind) option on the same command. Subroutine WRTCHK writes the actual checkpoint in the manner discussed in section I.B.3.

CHAPTER II
SUBROUTINE DOCUMENTATION

A. INTRODUCTION

The software details of the subroutines comprising the four modules of version 3 of the GEMACS code are given in this chapter. Common blocks and common block variable definitions are given in chapter III.

B. CONVENTIONS

There are several conventions of units and coordinate systems used throughout the code. These are discussed below to avoid needless repetition throughout the software documentation.

1. Units

With the exception of the GTD physics routines, GEMACS employs the following system of units:

Length - meters
Time - minutes
Frequency - megahertz
Electric field - volts/meter
Magnetic field - amps/meter
Input impedance - ohms
Interaction impedance - ohms/meter
Lumped load - ohms/meter
Load power - watts
Antenna voltage - volts
Current - amperes

Angles are expressed in radians for calculations and in degrees for input and output. Separate variables are used for radian and degree units and are indicated as such in the documentation.

With the addition of GTD physics to GEMACS it was found that GTD calculations could be made more rapidly if all dimensions were normalized to wavelengths. Hence, in GTD routines we have:

Length - wavelengths

Frequency - not used

Time - not used

Wavelength - meters

Electric field - volts/wavelength

Magnetic field - amps/wavelength

The units conversion is made in subroutine GTDDRV (see figure II-1). GTDDRV takes the standard GEMACS units, converts them to GTD units, calls the GTD physics routines for fields, and converts field units back to standard GEMACS units prior to returning control to ZGTDRV.

2. Passive Sign Convention

All loads and impedances are assumed to be passive. That is, even though a wire segment may radiate energy due to its excitation, the element itself is absorbing energy from its source. This definition implies the sign convention shown in figure II-2. With this convention we may write

$$Z = +\frac{V}{I}. \qquad \text{(II-1)}$$

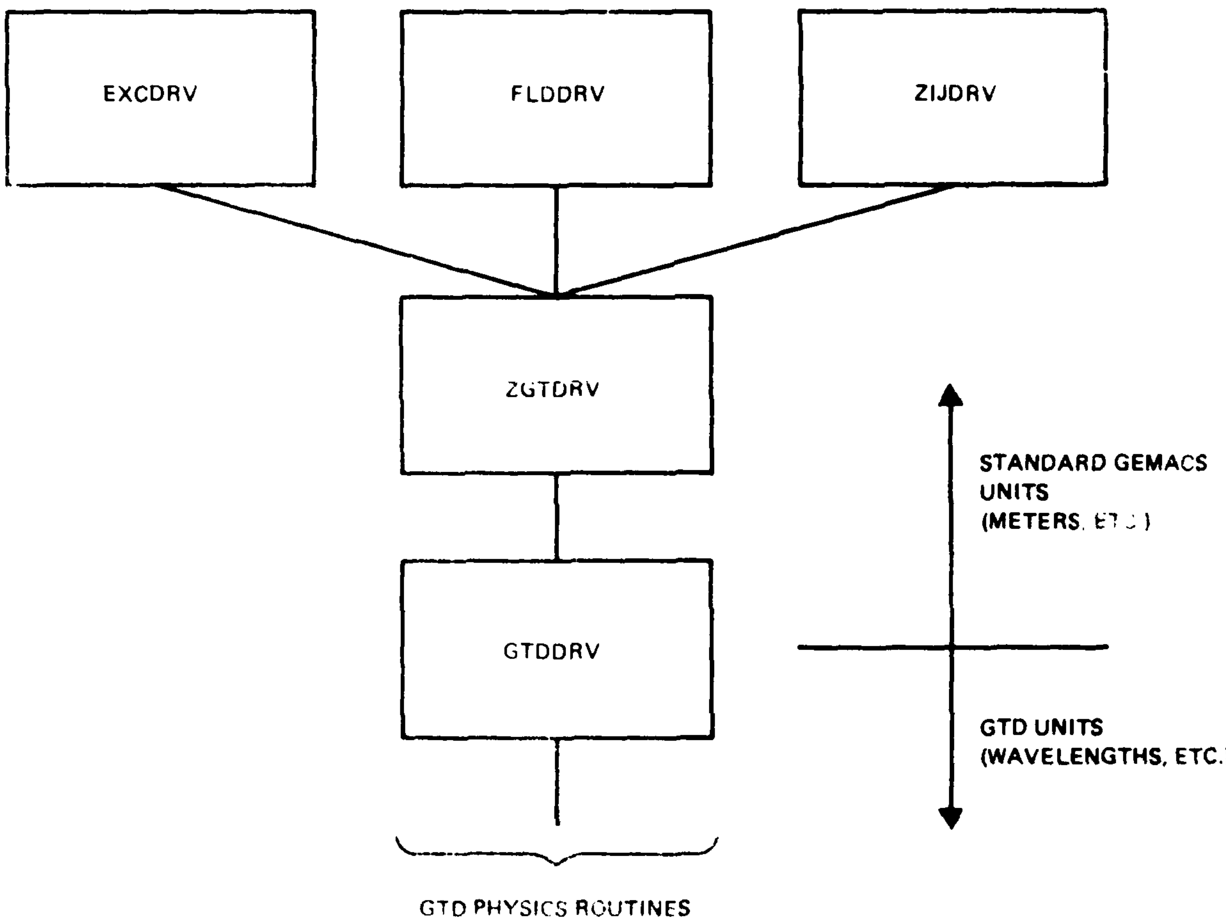
An understanding of this definition is important where loads are attached to wire segments. Consider the example in figure II-3. We know that

$$V_W = Z_W I_W$$

and

$$V_L = Z_L I_L + V_A. \qquad \text{(II-2)}$$

When the load is attached we know that $V_W = V_L$, but $I_L = -I_W$ because of the passive sign convention.



[illegible] Illustrating Where in the GEMACS GTD Module Units are Converted to and from Wavelengths, etc

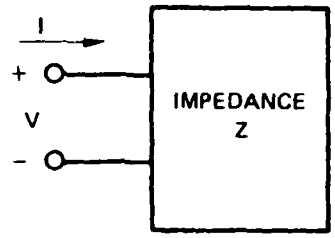


Figure II-2. The Passive Sign Convention

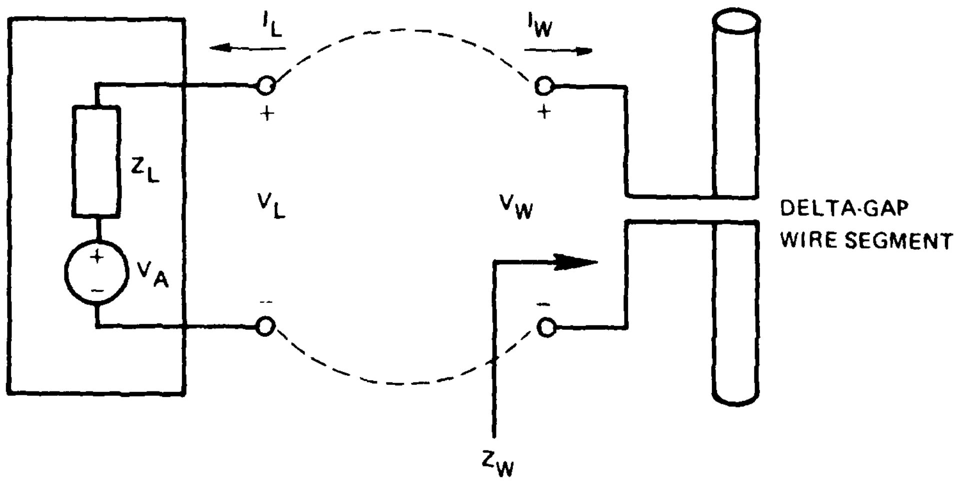


Figure II-3. Attaching a Load to a Wire Segment

When one forms the MOM matrix problem he obtains

$$[Z]\ [I] = [-V] \tag{II-3}$$

But our $[V]$ is represented by $\left[Z_L I_L + V_A\right] = \left[-Z_L I + V_A\right]$.

Hence, we obtain

$$\left[Z - Z_L\right]\left[I\right] = \left[-V_A\right]. \tag{II-4}$$

It is the passive sign convention that is responsible for the negative sign in (II-4).

3. <u>Coordinate Systems</u>

While one global coordinate system is used throughout the majority of GEMACS, several auxiliary systems are used to ease the computation of GTD interactions. The principal ones are discussed below.

a. <u>GTD Reference Coordinate System (RCS)</u>

The reference coordinate system is the fundamental system for GTD calculations. The system geometry is defined and stored in the reference coordinate system. Many of the calculations carried out are done in the reference coordinate system. Each of the other coordinate systems is defined in terms of this system.

When only plates are present in the GTD geometry, the RCS is identical to the global coordinate system. The presence of a cylinder causes the RCS to be rotated so that its z-axis lies along the axis of the cylinder and the x- and y-axes correspond to the major and minor axes of the cylinder cross section, respectively. The origin of the RCS is translated to the center of the cylinder. See Figures [illegible] and 2 [illegible] [illegible] subroutine CYAXIS writeup.

b. <u>Source Coordinate System</u>

The source coordinate system is the system in which the orientation of the source is defined. There is one such system for each source, although only one appears in the computations at a given time.

Each time another source is used the source coordinate system is redefined.

A one-dimensional source, such as a dipole or wire segment, lies along the z-axis of the source coordinate system. A patch lies in the (x-y) plane. Spherical wave sources are centered in the source coordinate system, and $\hat{\theta}$ and $\hat{\Phi}$ field polarizations correspond to the $\hat{\theta}$ and $\hat{\Phi}$ directions in that system. The same is true for plane wave sources. The software description for subroutine SOURCE has figures which illustrate these coordinate systems.

c. Source Image Coordinate System

In many cases the code uses image theory in computing fields reflected from a perfectly conducting plate or cylinder end cap. This involves computing an image source from which the reflected rays will appear to originate (see section on subroutine REFPLA). Assuming the source dimensions are known, the image source may be determined by computing the source image location using subroutine IMAGE and the source image orientation using subroutine IMDIR for a plate and IMCDIR for an end cap. As the source location and orientation are specified using a source coordinate system, the "source image coordinate system" is used to define the image source. The location of the source image is the origin of the source image coordinate system, and the axes are defined by unit vectors in the same manner as for the source coordinate system.

d. Edge - Fixed Coordinate System

The code generates an edge - fixed coordinate system for each edge on every plate. The three rectangular coordinate system axes for each edge are positioned as follows:

(a) In the plate plane and normal to the edge (the edge binormal)

(b) Normal to the plate

(c) Along the plate edge.

Figure 1 of subroutine DIFPLT illustrates the placement of the vectors.

The most significant use of the edge-fixed system is in determining edge diffractions. Incident and diffracted ray propagation angles along with polarization vectors are calculated by taking dot and cross products of edge-fixed unit vectors and the ray propagation and polarization unit vectors. Edge-fixed unit vectors are also used in calculating geometry for intersecting plates, as well as checking to see if plates are flat. The edge-fixed vectors are calculated in section 2 of subroutine GEOM. Also, note that a similar set of vectors is used in calculating the diffracted fields from the rims of the cylinder end caps.

e. End Cap Diffraction Point Coordinate System

The code defines a special coordinate system at the point of end cap diffraction so that the incident field may be decomposed into parallel (end cap diffraction point coordinate system z-axis) and perpendicular components. The origin of the system is the point of diffraction. The z-axis is aligned with the end cap rim tangent. The x-axis is perpendicular to the rim tangent in the plane of the end cap. The y-axis is formed by $\hat{z} \times \hat{x}$. Figure 1 of subroutine ENDIF illustrates this coordinate system.

C. SOFTWARE DESCRIPTIONS

The description of each subroutine within GEMACS is given in this section. The subroutines are listed alphabetically with the exception of the main program and the block data subroutine, which are listed first. Each description consists of the name of the subroutine, the module(s) in which it is located, its purpose, a brief summary of its operation, the internal and external variables associated with the subroutine, the names of the subroutines calling that subroutine, and the names of the subroutines it calls. A flow diagram completes the description.

1. NAME: GEMACS (INPUT)

2. PURPOSE: MAIN Driver Program

3. METHOD: GEMACS directs execution through input processing (INPDRV), task execution (TSKXQT) and run termination (STATFN) processors in that sequence. If the input processor determines that the job is a restart-from-checkpoint job, task execution is bypassed if restart was specified for another module.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
IDAY	Date in mm/dd/yy format (A6, A2)
INDKWN	Keyword index to the NCODES array
INDX	Index to the ITEMP array
ITIME	Time of day in hh. mm format (24-hour clock)
KWMXP1	KWMAX plus 1
LSTSYS(20)	Contains pointer to KWNAME array for module-directed restart
MODCOD	Pointer to module name in KWNAME array
MODNAM	Module name variable
NAMMOD	Hollerith (A6) name of this module
NPRPRT	Number of keywords per line of output
NXT	Pointer to next keyword
NXTTMP	Pointer to next keyword in ITEMP
N2	Minimum number of keywords
RSTART	Restart flag
TIME	Equivalenced to ITIME

5. I/O VARIABLES:

A. INPUT LOCATION

INPUT	LOCATION
ISOFF	/ADEBUG/
ISON	/ADEBUG/
KWMAX	/PARTAB/
KWNAME	/PARTAB/
NCODES	/PARTAB/
NOGOFG	/SCNPAR/

B. OUTPUT: None

6. CALLING ROUTINES: Not applicable

7. CALLED ROUTINES:

ASSIGN

CONVRT

INPDRV

SHELL

STATFN

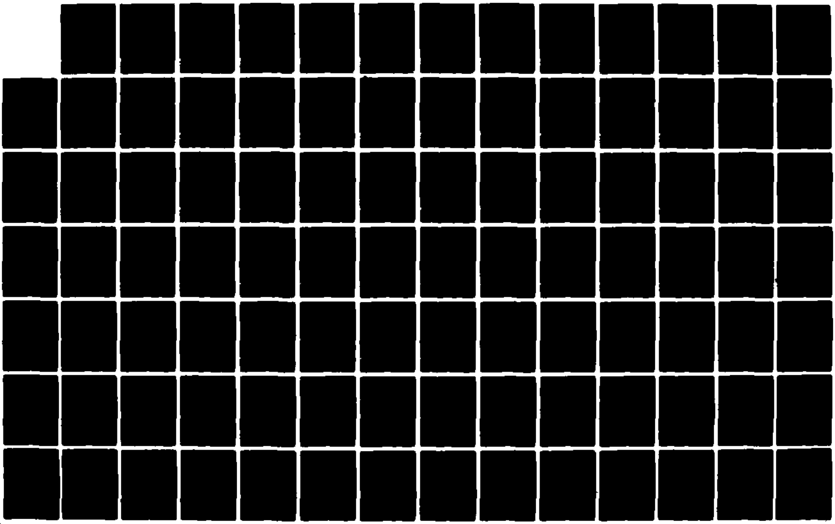
SYSRTN

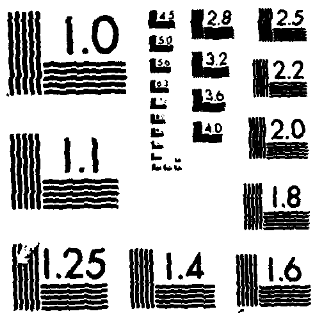
TSKXQT

AD-A137 463 GENERAL ELECTROMAGNETIC MODEL FOR THE ANALYSIS OF COMPLEX SYSTEMS (GEMACS..(U) BDM CORP ALBUQUERQUE NM D L KADLEC ET AL. SEP 83 BDM/A-83-020-TR

2/5

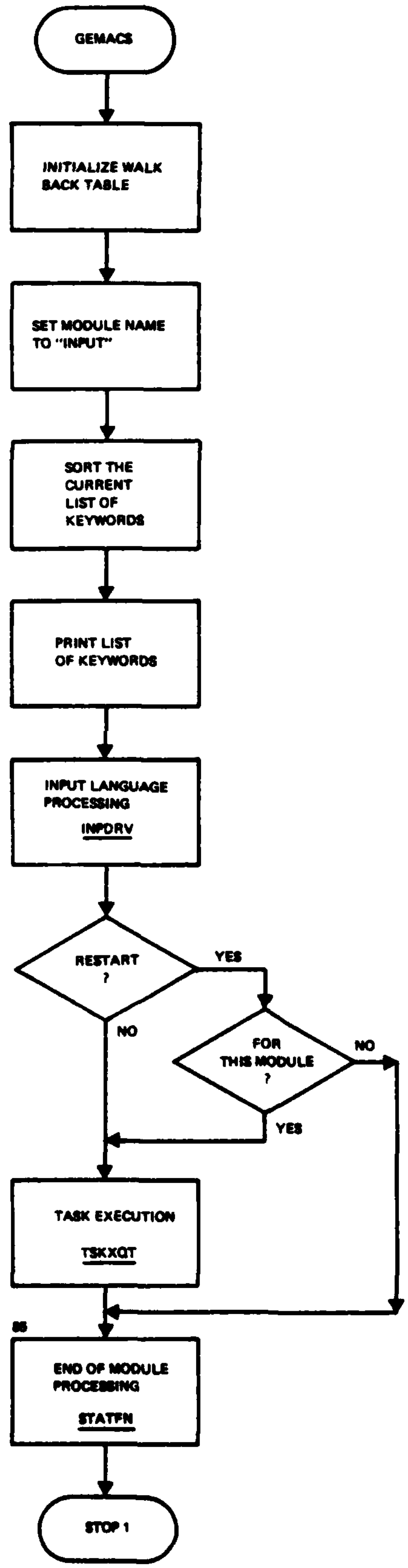
UNCLASSIFIED RADC-TR-83-217-VOL-3 F30602-81-C-0084 F/G 20/14 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

GEMACS (INPUT)



1. NAME: GEMACS (GTD, MOM, OUTPUT)

2. PURPOSE: Main driver program

3. METHOD: GEMACS directs execution through the module start-up (STRTUP), task execution (TSKXQT) and run termination (STATFN) processors in that sequence. If (1) STRTUP determines that the job is a restart-from-checkpoint job, (2) the restart flag RSTART is on, and (3) restart was specified for another module, STRTUP terminates module execution.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
IDAY	Date in mm/dd/yy format (A6, A2)
ITIME	Time of day in hh. mm format (24-hour clock)
MODNAM	Module name variable
NAMMOD	Hollerith (A6) name of this module: 6HGTD , 6HMOM , or 6HOUTPUT
TIME	Equivalenced to ITIME

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
ISOFF	/ADEBUG/
ISON	/ADEBUG/
NOGOFG	/SCNPAR/

B. OUTPUT: None.

6. CALLING ROUTINES: Not applicable

7. CALLED ROUTINES:

ASSIGN

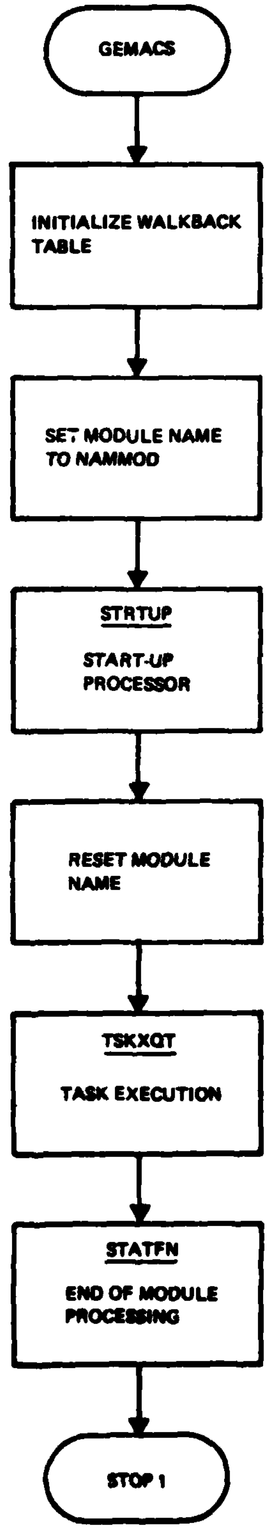
STATFN

STRTUP

SYSRTN

TSKXQT

GEMACS (GTD, MOM, OUTPUT)



1. NAME: BLOCK DATA BLKDAT (GTD, INPUT, MOM, OUTPUT)

2. PURPOSE: Load data into named common areas.

3. METHOD: Not applicable.

4. INTERNAL VARIABLES: See common block glossary, chapter III.

5. I/O VARIABLES: Not applicable.

6. CALLING ROUTINES: Not applicable.

7. CALLED ROUTINES: Not applicable.

1. NAME: ASSIGN (GTD, INPUT, MOM, OUTPUT)

2. PURPOSE: To enter the called subroutine names into the subroutine name table.

3. METHOD: After each subroutine is called, ASSIGN is called to insert the subroutine name in the NRNAMS table.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
NAMSB	Subroutine name
NUMSB	Assigned subroutine number
NUMSUB	Subroutine counter

5. I/O VARIABLES:

A. INPUT — LOCATION

INPUT	LOCATION
NAMSB	F.P.

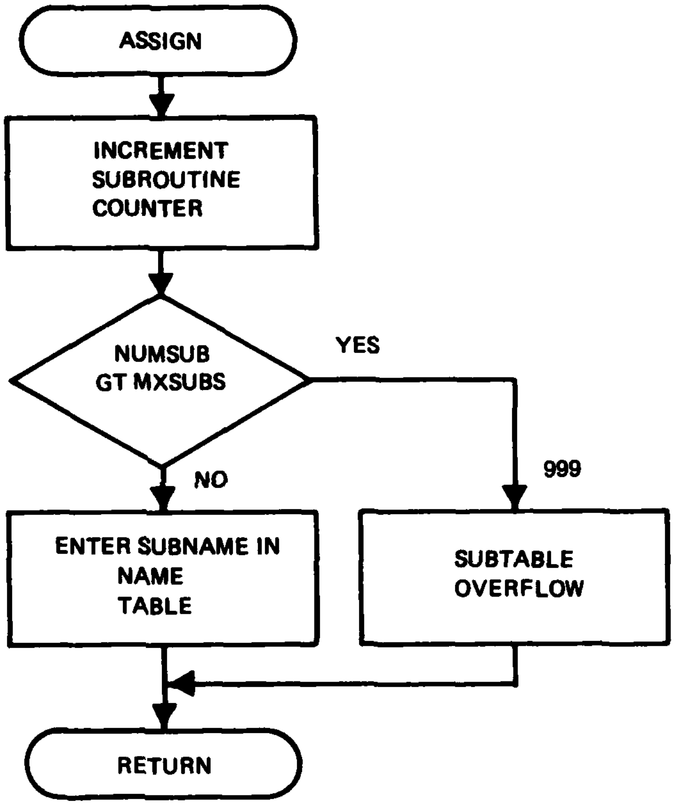
B. OUTPUT — LOCATION

OUTPUT	LOCATION
NUMSB	F.P.

6. CALLING ROUTINES: All major subroutines.

7. CALLED ROUTINES: None.

ASSIGN (GTD, INPUT, MOM, OUTPUT)



1. NAME: BABS (GTD)

2. PURPOSE: This function computes the absolute value of a complex argument. It is similar to CABS, except it avoids run time errors when the real part and imaginary part of the argument are zero.

3. METHOD: The system function CABS is used unless the absolute value of the real part and the imaginary part of the argument are close to zero, in which case a very small value is returned.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
BABS	Absolute value of the complex argument
X	Absolute value of the real part of Z
Y	Absolute value of the imaginary part of Z
Z	The complex argument

5. I/O VARIABLES:

A. INPUT	LOCATION
Z	F.P.

B. OUTPUT	LOCATION
BABS	FUNCTION

6. CALLING ROUTINES:

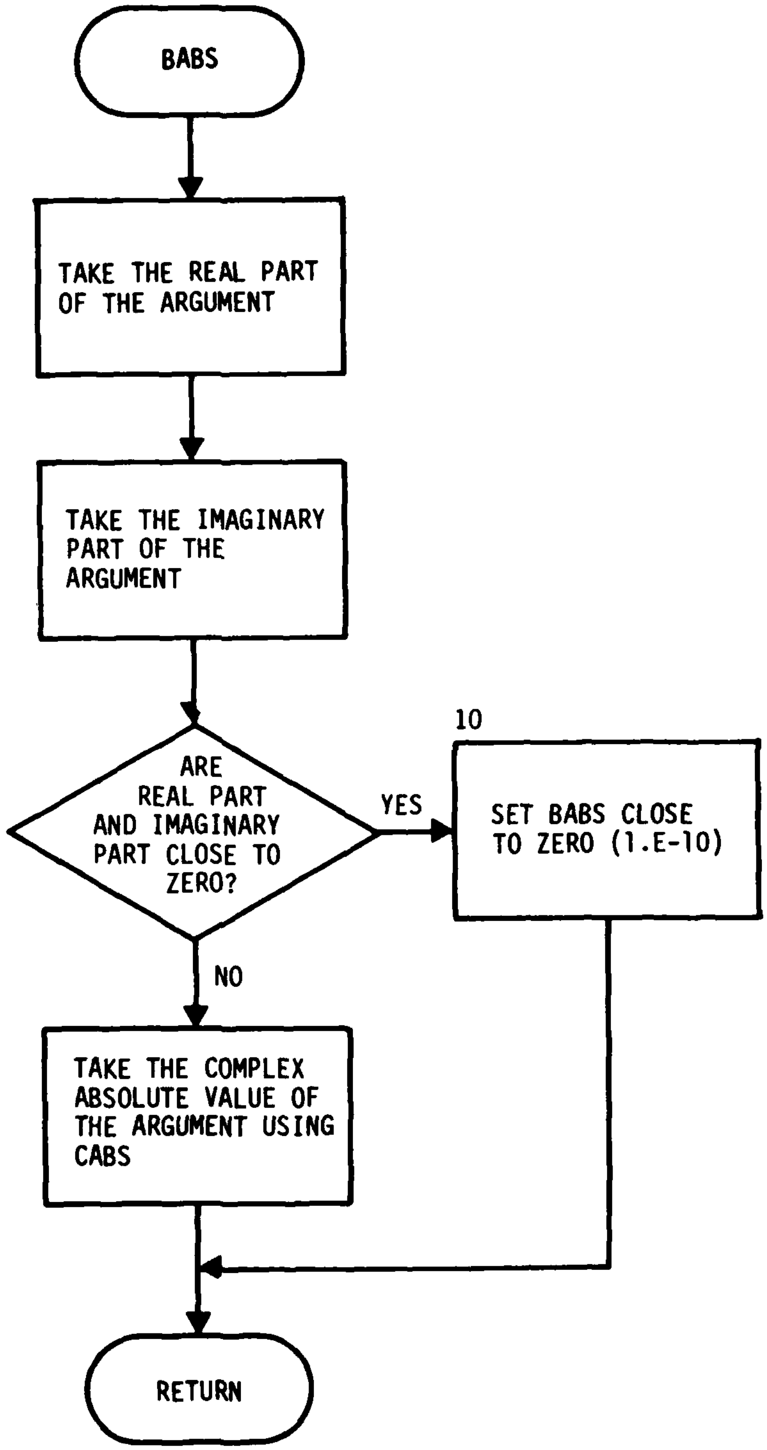
DFPTCL

DICOEF

POLYRT

7. CALLED ROUTINE: None.

BABS (GTD)



BABS
TAKE THE REAL PART OF THE ARGUMENT
TAKE THE IMAGINARY PART OF THE ARGUMENT
ARE REAL PART AND IMAGINARY PART CLOSE TO ZERO?
YES
10
SET BABS CLOSE TO ZERO (1.E-10)
NO
TAKE THE COMPLEX ABSOLUTE VALUE OF THE ARGUMENT USING CABS
RETURN

1. NAME: BACSUB (MOM)

2. PURPOSE: Execute the command BACSUB A * X = B by setting up the data to solve a set of simultaneous linear equations after matrix A has been decomposed.

3. METHOD: Retrieve command arguments, determine core available, load appropriate parts of the decomposed matrix and right-hand side into common area TEMP and call appropriate solution algorithm.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
IBAND	Flag set to 1 if matrix is banded
IFWD	Flag set to 0 for forward elimination and to 1 for back substitution
ILOWER	Flag set to 1 to indicate a lower triangular matrix
INDXA	Location of symbol A in NDATBL
INDXL	Location of lower triangular matrix decomposed from A in NDATBL
INDXU	Location of upper triangular matrix decomposed from A in NDATBL
IORDER	Flag set to 1 for a transposed matrix
IUPPER	Flag set to 1 for the upper triangular matrix
LOCA	Logical unit number for symbol A
LOCAIJ	Location of first word for symbol A
LOCL	Logical unit number for lower triangular matrix
LOCU	Logical unit number of upper triangular matrix
LSTNAM	Name of the last symbol used

BACSUB (MOM)

NAMEA	Symbolic name of decomposed matrix
NAMGET	Temporary location of upper and lower matrix names
NAMLWR	Symbolic name of lower triangular matrix
NAMUPR	Symbolic name of upper triangular matrix
NBITA	Attribute word for symbol A
NBITS	Attribute word for a data set
NCOLL	Number of columns in lower triangular matrix
NCOLU	Number of columns in upper triangular matrix
NM1	Submatrix number minus one
NPRELM	Number of words per element
NROWL	Number or rows in lower triangular matrix
NROWS	Number of row elements in core
NROWU	Number of rows in upper triangular matrix
NSYMBL	Number of entries in NDATBL array
NTCELL	Number of TEMP locations available for matrix storage
NTLEFT	Number of TEMP cells remaining to be used
NUMCOL	Number of columns in symbol A
NUMMAT	Number of submatrices in symbol A
NUMROW	Number of rows in symbol A
NXTCOL	Pointer to next column to be read into TEMP
N1	Column index of first column in core
N2	Column index of last column in core
RHS	Right-hand side solution

5. I/O VARIABLES:

A. INPUT — LOCATION

INPUT	LOCATION
A	F.P.
DBGPRT	/ADEBUG/
INDXA	F.P.
NDATBL	/PARTAB/
NM1	F.P.
NPDATA	/PARTAB/
NTCELL	F.P.
NUMMAT	F.P.
RHS	F.P.

B. OUTPUT — LOCATION

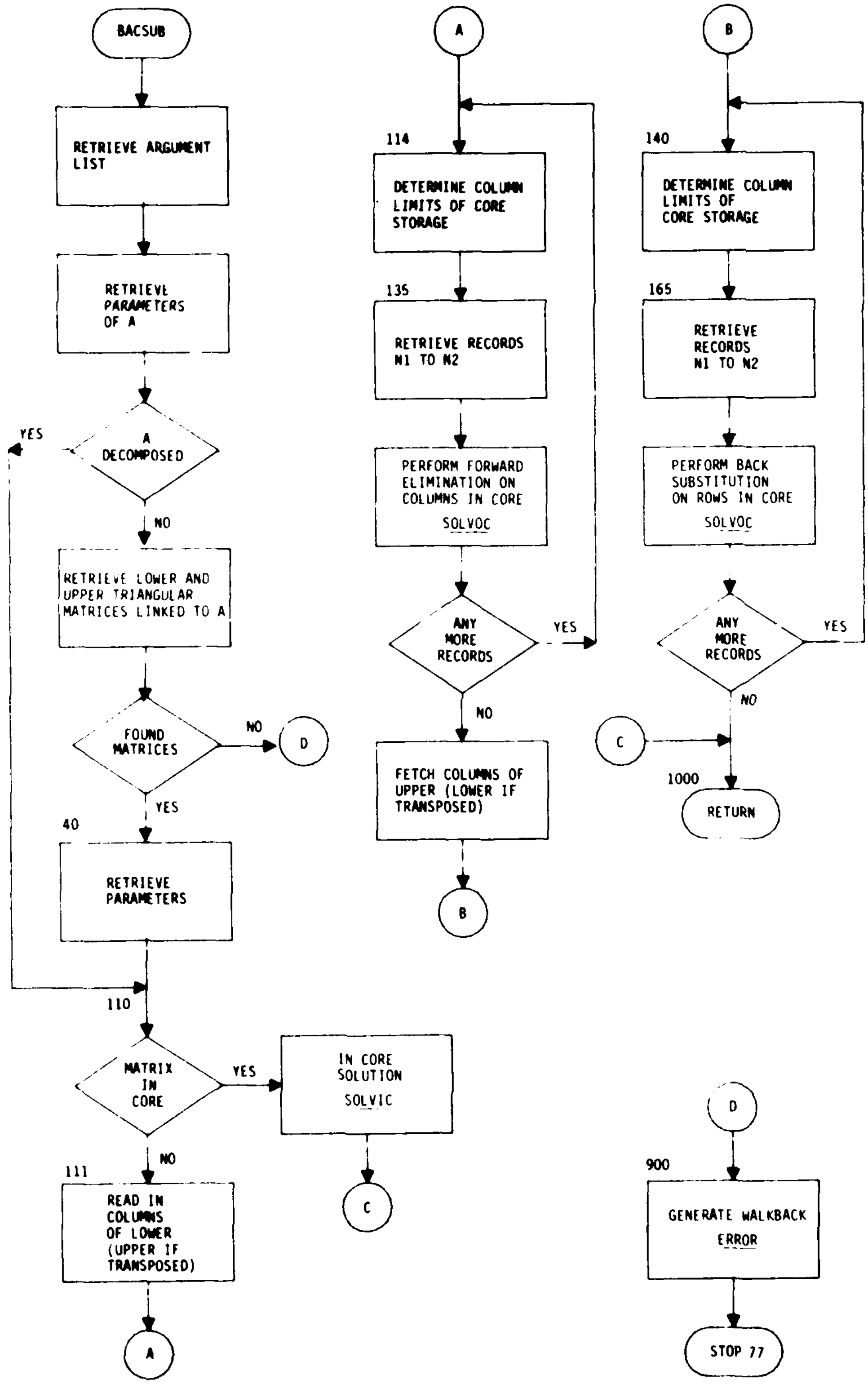
OUTPUT	LOCATION
RHS	F.P.

6. CALLING ROUTINES: SOLDRV

7. CALLED ROUTINES:

ASSIGN	IBITCK	STATOT
CONVRT	SOLVIC	WLKBCK
ERROR	SOLVOC	
GETSYM	STATIN	

BACSUB (MOM)


BACSUB
RETRIEVE ARGUMENT LIST
RETRIEVE PARAMETERS OF A
A DECOMPOSED
YES
NO
RETRIEVE LOWER AND UPPER TRIANGULAR MATRICES LINKED TO A
FOUND MATRICES
NO
D
YES
40
RETRIEVE PARAMETERS
110
MATRIX IN CORE
YES
IN CORE SOLUTION
SOLVIC
C
NO
111
READ IN COLUMNS OF LOWER (UPPER IF TRANSPOSED)
A
A
114
DETERMINE COLUMN LIMITS OF CORE STORAGE
135
RETRIEVE RECORDS N1 TO N2
PERFORM FORWARD ELIMINATION ON COLUMNS IN CORE
SOLVOC
ANY MORE RECORDS
YES
NO
FETCH COLUMNS OF UPPER (LOWER IF TRANSPOSED)
B
B
140
DETERMINE COLUMN LIMITS OF CORE STORAGE
165
RETRIEVE RECORDS N1 TO N2
PERFORM BACK SUBSTITUTION ON ROWS IN CORE
SOLVOC
ANY MORE RECORDS
YES
NO
C
1000
RETURN
D
900
GENERATE WALKBACK
ERROR
STOP 77

1. NAME: BANDIT (MOM)

2. PURPOSE: Execute the command: B = BAND (A), BNDW = N

3. METHOD: Extract the N elements above and below the diagonal element in each column of A and store the result as symbol B.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
BNDMAG	Magnitude of the banded elements extracted
COLMAG	Magnitude of all column elements
FRSTIM	Logical TRUE if first record of submatrix
IBAND	Number of elements above and below diagonal element to be kept in the band
IN1	Location of A in the symbol table
IN2	Location of the B in the symbol table
IN2BND	Banded attribute for B
IROWS	Number of rows A will have
IROW1	Location of row 1 of submatrix N in B
MORE	Exponent of 2 if complex or double precision
NAME1	Symbolic name of A
NAME2	Symbolic name of B
NBIT1	Attribute word for A
NBIT2	Attribute word for B
NCN	Loop index
NCOLS	Number of columns in B
NPRBND	Number of words per band
NPRCOL	Number of words per column

NPRELM	Number of words per element
NROWS	Number of rows in B
NUMMAT	Number of submatrices
N1	Temporary location for the name of A
N2	Temporary location for the name of B
RATIO	Ratio of BNDMAG to COLMAG

5. I/O VARIABLES:

A. INPUT — LOCATION

INTARG	/ARGCOM/
IPASS	/ARGCOM/
NDATBL	/PARTAB/

B. OUTPUT — LOCATION

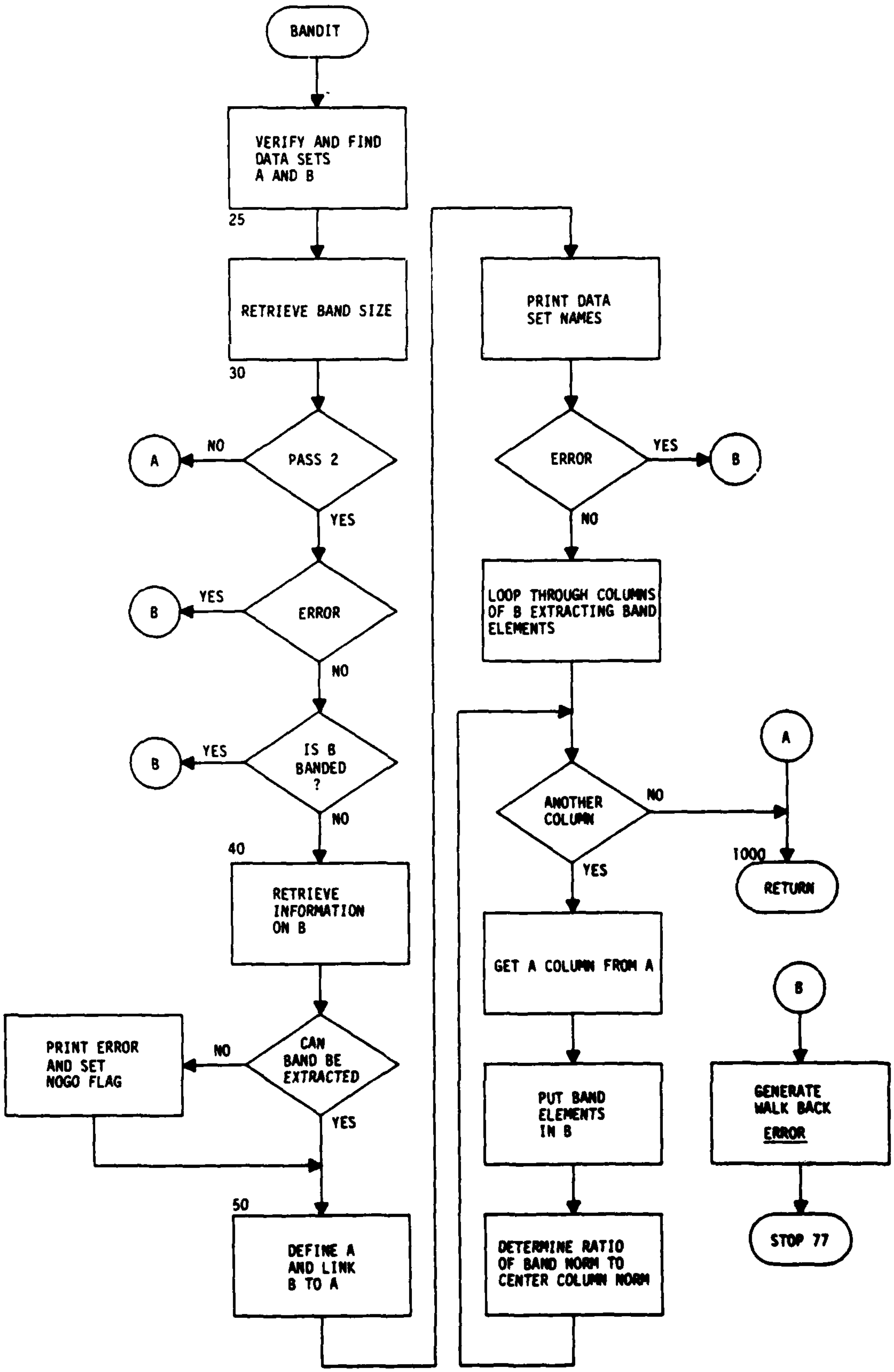
NDATBL	/PARTAB/

6. CALLING ROUTINES: TSKXQT

7. CALLED ROUTINES:

ASSIGN	IBITCK	SYMDEF
CONVRT	PUTSYM	SYMUPD
ERROR	STATIN	WLKBCK
GETSYM	STATOT	

BANDIT (MOM)


BANDIT
VERIFY AND FIND DATA SETS A AND B
25
RETRIEVE BAND SIZE
30
PASS 2
NO
A
YES
ERROR
YES
B
NO
IS B BANDED ?
YES
B
NO
40
RETRIEVE INFORMATION ON B
CAN BAND BE EXTRACTED
NO
PRINT ERROR AND SET NOGO FLAG
YES
50
DEFINE A AND LINK B TO A
PRINT DATA SET NAMES
ERROR
YES
B
NO
LOOP THROUGH COLUMNS OF B EXTRACTING BAND ELEMENTS
ANOTHER COLUMN
NO
A
1000
RETURN
YES
GET A COLUMN FROM A
PUT BAND ELEMENTS IN B
DETERMINE RATIO OF BAND NORM TO CENTER COLUMN NORM
B
GENERATE WALK BACK ERROR
STOP 77

1. NAME: BEXP (GTD)

2. PURPOSE: To determine the exponential of a complex argument. It is similar to the library function CEXP, but it avoids the problem of the argument having a large imaginary part.

3. METHOD: The argument is separated into its real and imaginary parts. The imaginary part is reduced to an equivalent angle between zero and 2π, including zero. Then the argument is put back together into a complex number and the CEXP function is performed on it.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
ARG	Exponential argument
ARGI	Imaginary part of argument
ARGII	Imaginary part of argument reduced to a number between or equal to 0 and 2π
ARGR	Real part of argument
BEXP	exp(ARG) = exp(ARGR + jARGII)
CJ	Complex number (0., 1.) = $\sqrt{-1}$
PI	π

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
ARG	F.P.
CJ	/COMP/
PI	/PIS/

B. OUTPUT

OUTPUT	LOCATION
BEXP	FUNCTION

6. CALLING ROUTINES:

DICOEF

DIFPLT

DPI

DPLRCL

DPLRPL

DZCOEF

ENDIF

FFCT

FKY

INCFLD

PFUN

QFUN

RCLDPL

RCLRPL

REFCAP

REFCYL

REFPLA

RPLDPL

RPLRCL

RPLRPL

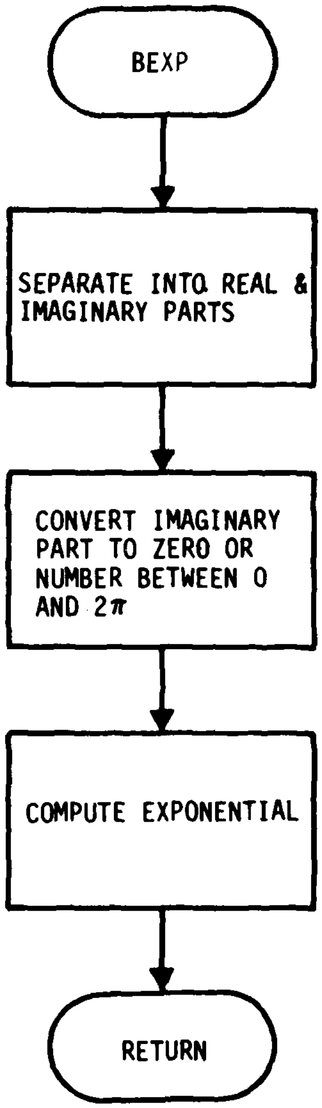
RPLSCL

SCLRPL

SCTCYL

7. CALLED ROUTINE: None.

BEXP (GTD)


BEXP
SEPARATE INTO REAL & IMAGINARY PARTS
CONVERT IMAGINARY PART TO ZERO OR NUMBER BETWEEN 0 AND 2π
COMPUTE EXPONENTIAL
RETURN

1. NAME: BMIRHS (MOM)

2. PURPOSE: To accumulate the complex matrix sum and product R = R-A*S

3. METHOD: The matrix operation R = R-A*S may be done accumulatively when A will not fit in core. R and S are column vectors and A is a matrix which has had a band of bandwidth M extracted. The product of A*S may be written as:

$$r_K = r_K - \sum_{n=n_1}^{n_2} a_{nk} S_n$$

where n_1 and n_2 are the columns of A in core.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
A	Matrix argument
J	First column of A in core
JM	J-1
K	Actual column index of A
KRL	First row element of A after band
KRU	Last row element of A before band
M	Bandwidth
MP	M + 1
NC	Number of columns of A in core
NCM	First element of SOL after band
NR	Number of rows in SOL, RHS, and A
RHS	Matrix argument
RHSK	Temporary storage
SOL	Solution currents

5. I/O VARIABLES:

A. INPUT — LOCATION

INPUT	LOCATION
A	F.P.
J	F.P.
M	F.P.
NC	F.P.
NR	F.P.
RHS	F.P.
SOL	F.P.

B. OUTPUT — LOCATION

OUTPUT	LOCATION
RHS	F.P.

6. CALLING ROUTINES: SOLDRV

7. CALLED ROUTINES:

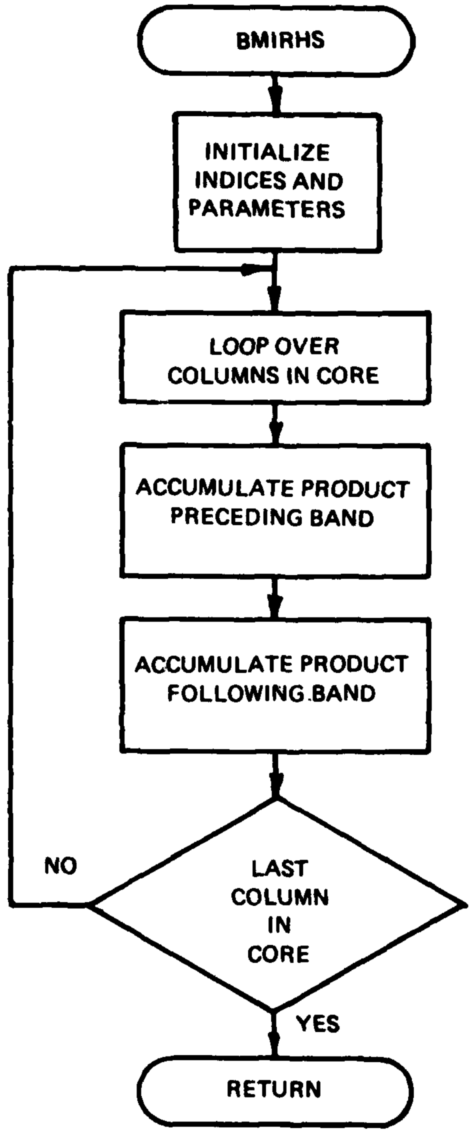
ASSIGN

STATIN

STATOT

WLKBCK

BMIRHS (MOM)


BMIRHS
INITIALIZE INDICES AND PARAMETERS
LOOP OVER COLUMNS IN CORE
ACCUMULATE PRODUCT PRECEDING BAND
ACCUMULATE PRODUCT FOLLOWING BAND
LAST COLUMN IN CORE
NO
YES
RETURN

1. NAME: BTAN2 (GTD)

2. PURPOSE: This function computes the two argument arctangent function. It is similar to ATAN2, except it avoids run time errors when the second argument is zero.

3. METHOD: The system function ATAN2(Y,X) is used to return the angle in radians, whose sine is Y and cosine is X unless the second argument or both of the arguments are zero. If the second argument is zero, either $\pi/2$ or $-\pi/2$ is returned depending on the sign of the first argument. If both arguments are zero, a zero value is returned.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
BTAN2	Two argument arctangent
X	Second argument, which is the cosine of the angle to be computed
Y	First argument, which is the sine of the angle to be computed

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
PI	/PIS/
X	F.P.
Y	F.P.
ZERO	/ADEBUG/

B. OUTPUT

OUTPUT	LOCATION
BTAN2	FUNCTION

6. CALLING ROUTINES:

CAPINT

CYLINT

DFPTWD

DIFPLT

DPLRCL

DPLRPL

ENDIF

GEOM

GEOMPC

GTDDRV

NFD

PLAINT

RCLDPL

RCLRPL

REFBP

REFCYL

RFDFIN

RFDFPT

RFPTCL

RPLDPL

RPLRCL

RPLSCL

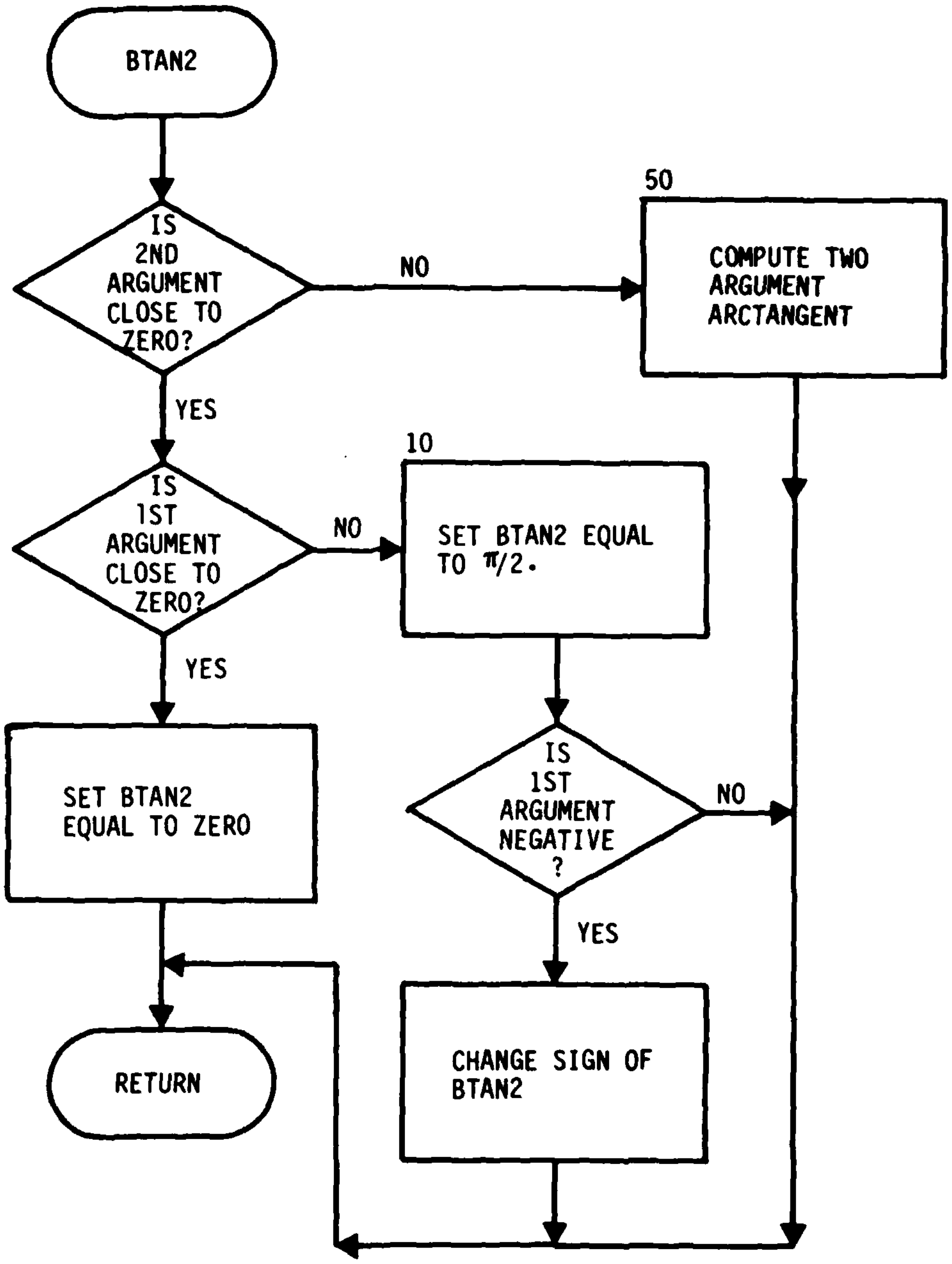
SCLRPL

SCTCYL

TANG

XYZFLD

7. CALLED ROUTINES: None.


BTAN2
IS 2ND ARGUMENT CLOSE TO ZERO?
NO
50
COMPUTE TWO ARGUMENT ARCTANGENT
YES
10
IS 1ST ARGUMENT CLOSE TO ZERO?
NO
SET BTAN2 EQUAL TO π/2.
YES
SET BTAN2 EQUAL TO ZERO
IS 1ST ARGUMENT NEGATIVE ?
NO
YES
CHANGE SIGN OF BTAN2
RETURN

1. NAME: BUBBLE (INPUT)

2. PURPOSE: BUBBLE does a bubble sort on the geometry data to assure that the wire segments and then the patch segments are sorted in ascending order in the SEGMNT array.

3. METHOD: This subroutine makes two sorts. First, it sorts the wire segments. If any wire segment after the i^{th} location in SEGMNT is found to have a smaller segment number, it and the i^{th} location are interchanged. After all wire segments have been sorted, the process is repeated in order to sort all the patch segments. Note that all wire segments always precede the patch segments.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
I	Counter for the data blocks
IBLK	Saved value of data block being filled
ILIM	Maximum number of segments in a data block
[illegible]	Outer loop index to SEGTBL array
IS	Segment number
ISGTBL	Equivalenced to SEGTBL
ITAG	Tag number for segment
IWPFG	Flag indicating patch sort
J	Segment number sought
LIMINR	Maximum number of segments in a data block for inner loop
[illegible]	Maximum number of segments in a data block for outer loop
MAXBLK	Maximum number of data blocks
MAXSEG	Maximum number of segments in each data block
NDXBLK	Data block in current use
[illegible]	New segment number

NNDEX	Inner loop index to SEGTBL array
NOGOFG	No go flag
NPATCH	Number of patches
NPRSEG	Number of data items per segment
NS	Counter of the number of segments
NUMSEG	The total number of wire segments and patches
NWIRE	Number of wire segments
SAVDAT	Temporary location to move segment data
SEGTBL	Array of segment data
TEMP	Temporary storage array

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
ISGTBL	/SEGMNT/
MAXBLK	/SEGMNT/
MAXSEG	/SEGMNT/
NDXBLK	/SEGMNT/
NPATCH	/SEGMNT/
NPRSEG	/SEGMNT/
NUMSEG	/SEGMNT/
NWIRE	/SEGMNT/
SEGTBL	/SEGMNT/

B. OUTPUT

OUTPUT	LOCATION
NOGOFG	/ADEBUG/
SEGTBL	/SEGMNT/

6. CALLING ROUTINE:

   GEODRV

7. CALLED ROUTINES:

   ASSIGN

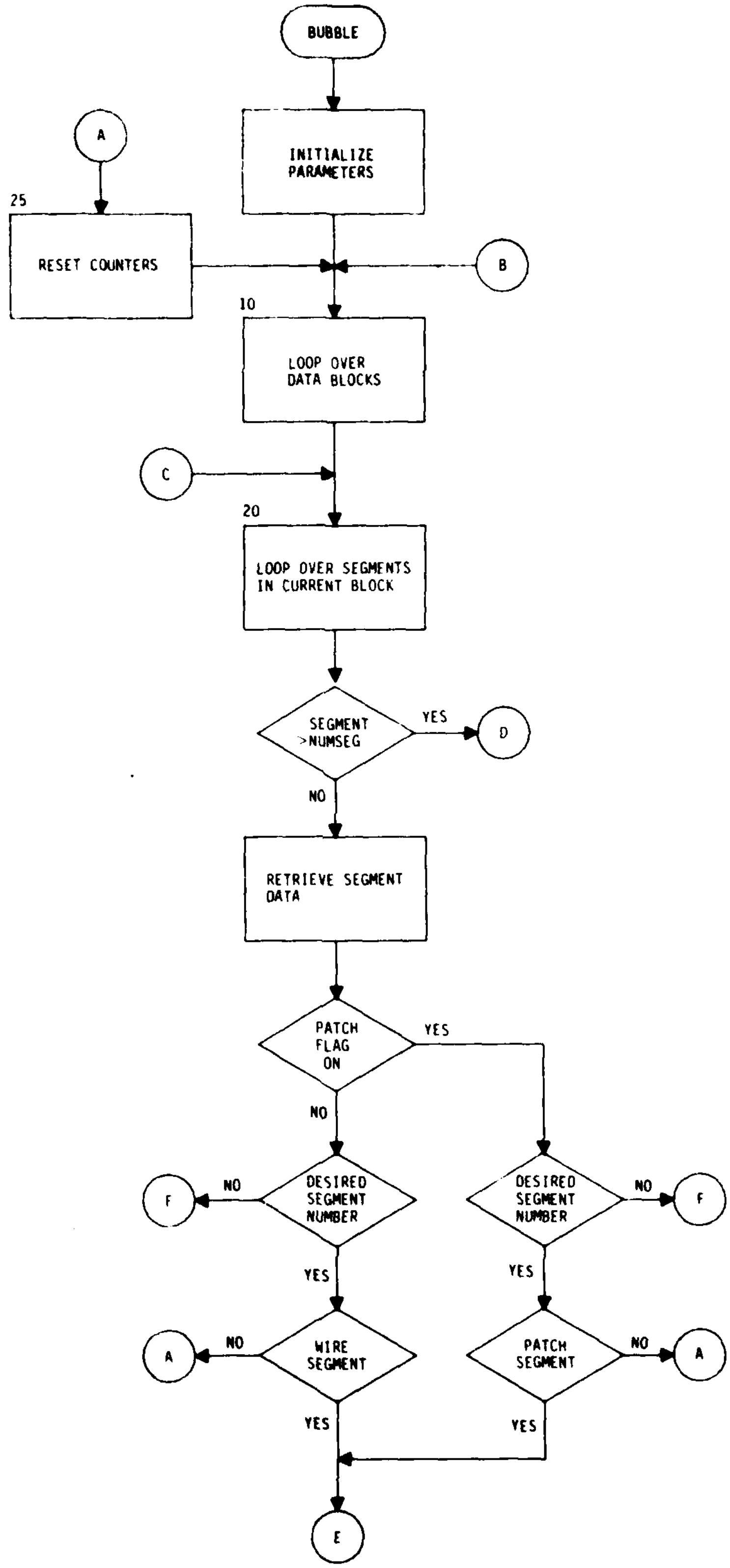
   GETSEG

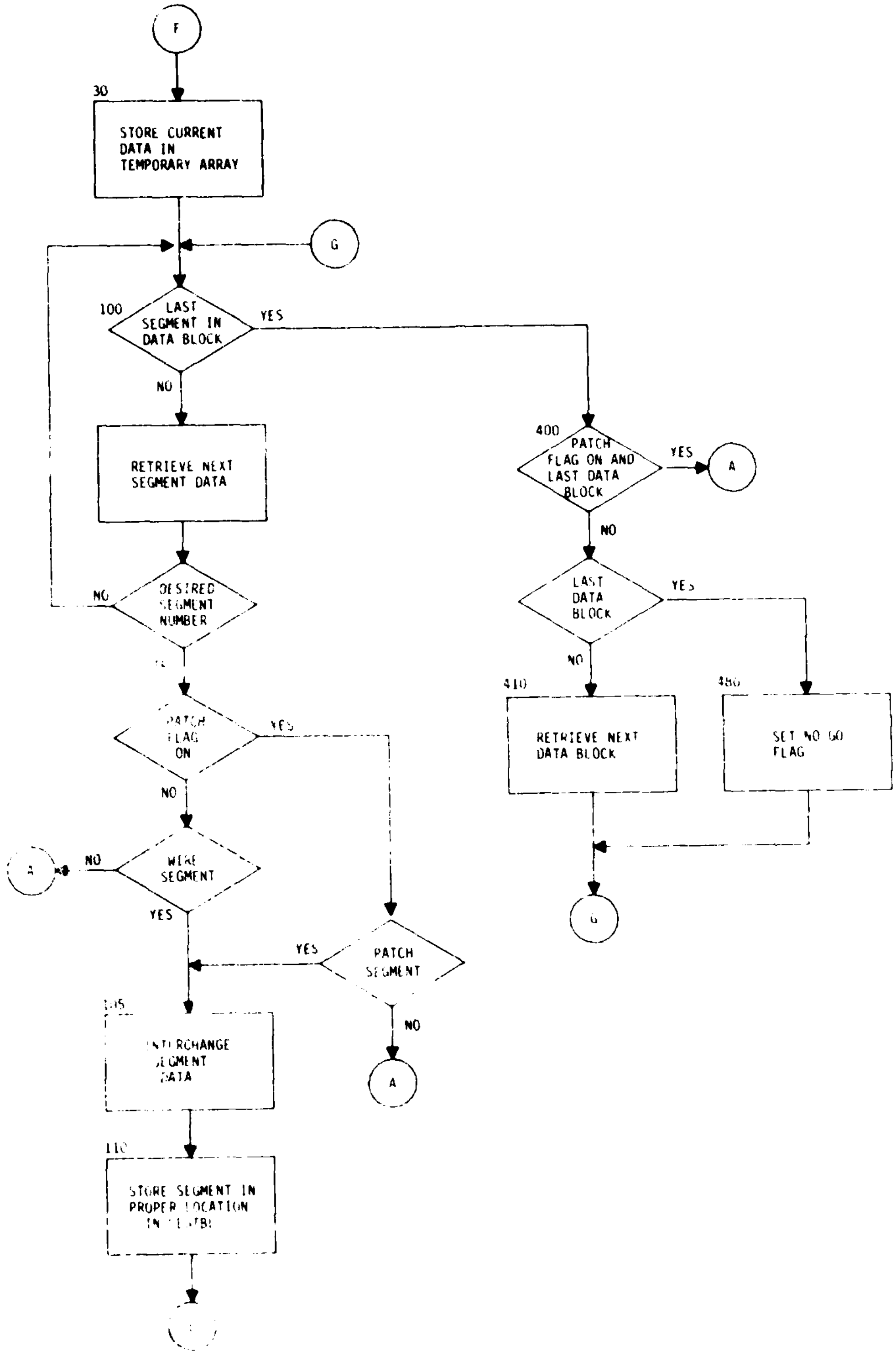
   STATIN

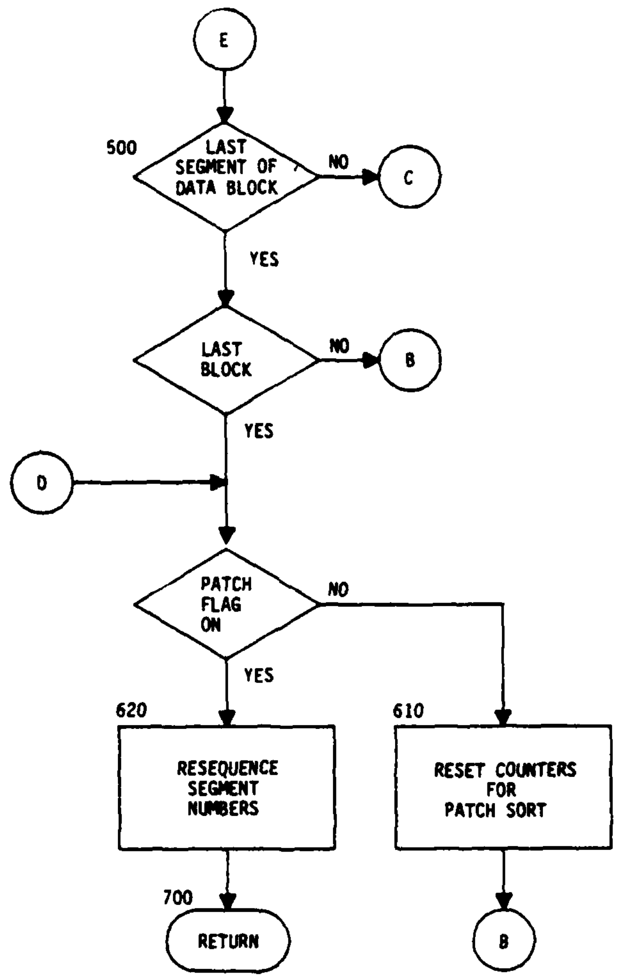
   STATOT

   WLKBCK

BUBBLE (INPUT)




F
30
STORE CURRENT DATA IN TEMPORARY ARRAY
G
100
LAST SEGMENT IN DATA BLOCK
YES
NO
RETRIEVE NEXT SEGMENT DATA
NO
DESIRED SEGMENT NUMBER
PATCH FLAG ON
YES
NO
A
NO
WIRE SEGMENT
YES
YES
PATCH SEGMENT
NO
A
105
INTERCHANGE SEGMENT DATA
110
STORE SEGMENT IN PROPER LOCATION
400
PATCH FLAG ON AND LAST DATA BLOCK
YES
A
NO
LAST DATA BLOCK
YES
NO
410
RETRIEVE NEXT DATA BLOCK
SET NO GO FLAG
G


E
500
LAST SEGMENT OF DATA BLOCK
NO
C
YES
LAST BLOCK
NO
B
YES
D
PATCH FLAG ON
NO
YES
620
RESEQUENCE SEGMENT NUMBERS
610
RESET COUNTERS FOR PATCH SORT
700
RETURN
B

1. NAME: CABC (MOM)

2. PURPOSE: To compute the coefficients in the sinusoidal basis functions for the current on each wire segment, given current at the center of each segment. For patches, the x, y, and z components of the surface current are determined assuming a pulse expansion function. These coefficients are needed for calculation of fields produced by the current distribution.

3. METHOD: The current basis function for wire segment i is

$$I_i(s) = A_i + B_i \sin k(s - s_i) + C_i \cos k(s - s_i)$$

where s represents path length along the segment and $s = s_i$ at the center of segment i. If ℓ and k are the segments before and after segment i, relative to the current reference direction of segment i, the continuity conditions

$$I_i(s_\ell) = I_\ell(s_\ell) \equiv I_\ell$$

$$I_i(s_k) = I_k(s_k) \equiv I_k$$

$$I_i(s_i) \equiv I_i$$

yield the coefficients

$$A_i = 1/\Delta \left[I_\ell \sin k\, \delta_{ik} - I_i \sin k\, (\delta_{i\ell} + \delta_{ik}) + I_k \sin k\, \delta_{i\ell}\right]$$

$$B_i = 1/\Delta \left[I_\ell(\cos k\, \delta_{ik} - 1) + I_i(\cos k\, \delta_{i\ell} - \cos k\, \delta_{ik}) + I_k(1 - \cos k\, \delta_{i\ell})\right]$$

$$C_i = -1/\Delta \left[I_\ell \sin k\, \delta_{ik} - I_i(\sin k\, \delta_{i\ell} + \sin k\, \delta_{ik}) + I_k \sin k\, \delta_{i\ell}\right]$$

$$\Delta = \sin k\,\delta_{i\ell} + \sin k\,\delta_{ik} - \sin k\,(\delta_{i\ell} + \delta_{ik})$$

where $\delta_{i\ell} = \left|s_i - s_\ell\right| \qquad \delta_{ik} = \left|s_k - s_i\right|$

$$k = 2\pi/\lambda$$

At a multiple junction segment ℓ or k is replaced by several segments connected to the junction. The corresponding δ is then the average of the distance from center of segment i to the center of each of the other segments, and I_ℓ or I_k is replaced by the algebraic sum of currents in the connected segments.

For each patch element, a pulse expansion function is assumed. For each patch, two currents J_1 and J_2 are determined. The spatial components of the surface current are found from:

$$J_x = J_1\,\hat{T}_{1x} + J_2\,\hat{T}_{2x}$$

$$J_y = J_1\,\hat{T}_{1y} + J_2\,\hat{T}_{2y}$$

$$J_z = J_1\,\hat{T}_{1z} + J_2\,\hat{T}_{2z}$$

where $\hat{T}_1$ and $\hat{T}_2$ are the orthogonal unit vectors which define the patch orientation in space.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
AX	A_1 for wires, J_x for patches
BX	B_1 for wires, J_y for patches
CELLO	Δ

CABC (MOM)

CK	$k\delta_{ik}$
CL	$k\delta_{i\ell}$
CLL	I_i
CLO	I_ℓ
CLY	I_k
COSK	$\cos(k\delta_{ik})$
COSL	$\cos(k\delta_{i\ell})$
CUR	Current at the center of the segment
CX	C_i for wires, J_z for patches
INCORE	Logical TRUE when AX, BX, CX are in core
IOSCR1	Temporary scratch file (Logical Unit 1)
JBLK	Block number for patch blocks
JIXK	Index of a segment directed into first end of segment i at multiple junction
JIZK	Index of a segment directed into second end of segment i at multiple junction
JLOC	Index in CUR array for patches
JOXK	Index of segment directed out of first end of segment i at multiple junction
JOZK	Index of segment directed out of second end of segment i at multiple junction
LAI	Location for the imaginary part of A
LAR	Location for the real part of A
LBI	Location for the imaginary part of B
LBR	Location for the real part of B
LCI	Location for the imaginary part of C

LCR	Location for the real part of C
SILK	$\sin\left[k(\delta_{i\ell} + \delta_{ik})\right]$
SINK	$\sin(k\delta_{ik})$
SINL	$\sin(k\delta_{i\ell})$

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
ANUMK	/ANUM/
ANUML	/ANUM/
CUR	F.P.
DBGPRT	/ADEBUG/
DIK	/AMPZIJ/
DIL	/AMPZIJ/
INCORE	F.P.
JCO1	/AMPZIJ/
JCO2	/AMPZIJ/
JIX	/JUNCON/
JIZ	/JUNCON/
JOX	/JUNCON/
JOZ	/JUNCON/
LOCAII	/FLDCOM/
LOCAIR	/FLDCOM/
LOCBII	/FLDCOM/
LOCBIR	/FLDCOM/
LOCCII	/FLDCOM/
LOCCIR	/FLDCOM/

NCIX	/JUNCON/
NCIZ	/JUNCON/
NCOX	/JUNCON/
NCOZ	/JUNCON/
NUMSEG	/SEGMNT/
NWIRE	/SEGMNT/

B. OUTPUT LOCATION

TEMP	/TEMP01/

6. CALLING ROUTINE:

FLDDRV

7. CALLED ROUTINES:

ASSIGN

CLSFIL

GETSEG

OPNFIL

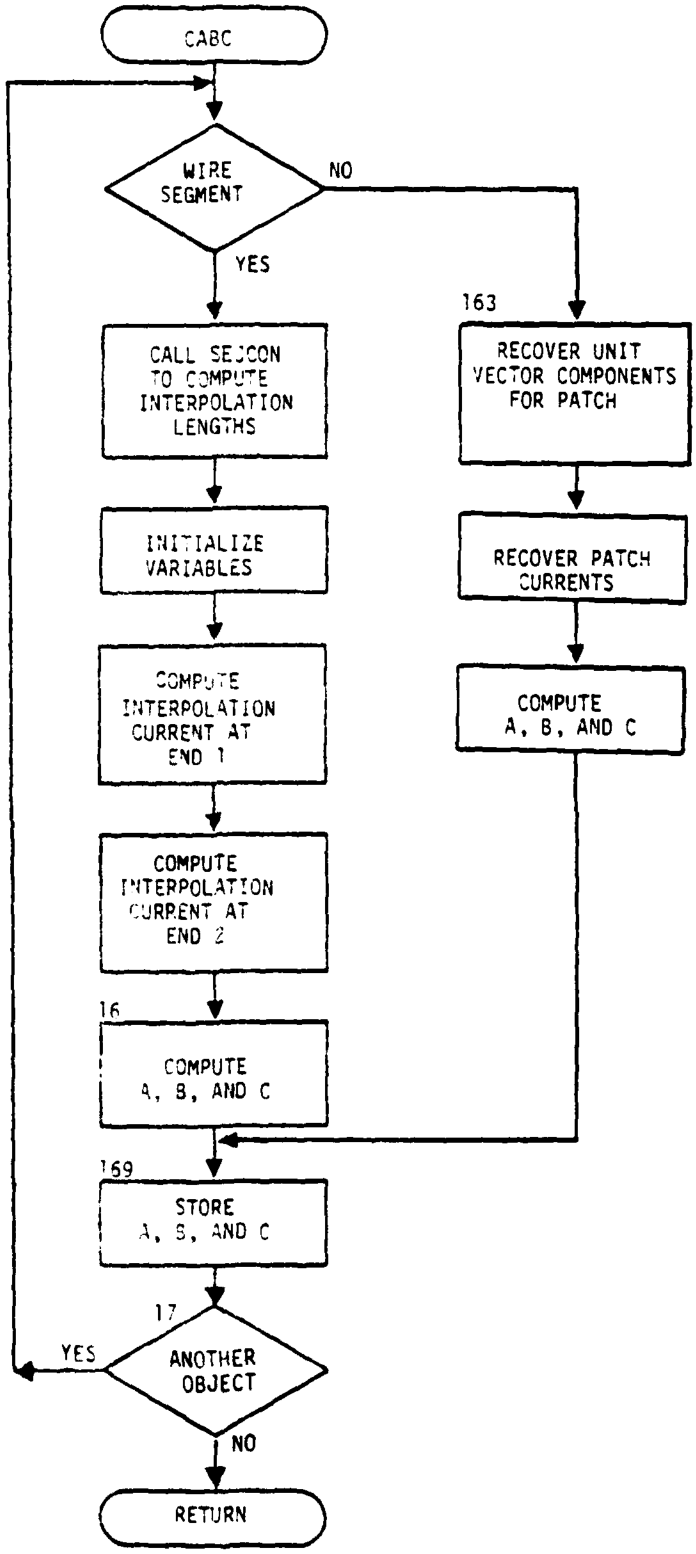
SEJCON

STATIN

STATOT

WLKBCK

CABC (MOM)



CABC
WIRE SEGMENT
NO
YES
163
RECOVER UNIT VECTOR COMPONENTS FOR PATCH
RECOVER PATCH CURRENTS
COMPUTE A, B, AND C
CALL SEJCON TO COMPUTE INTERPOLATION LENGTHS
INITIALIZE VARIABLES
COMPUTE INTERPOLATION CURRENT AT END 1
COMPUTE INTERPOLATION CURRENT AT END 2
16
COMPUTE A, B, AND C
169
STORE A, B, AND C
17
ANOTHER OBJECT
YES
NO
RETURN

1. NAME: CAPINT (GTD)

2. PURPOSE: To determine if a ray traveling from a given source location in a given direction will hit a cylinder end cap.

3. METHOD: The subroutine checks to see if a ray emanating from a source in a given direction hits a cylinder end cap. First it checks if the ray is aimed toward or away from the end cap plane by comparing the sign of the dot product of the scatter direction and end cap normal (DN) with the sign of the dot product of the source location vector and end cap normal (AN). If the ray is directed toward the end cap plane as shown in figure 1, the intersection point with the plane is found from:

$$\overline{XT} = \overline{XIS} - \hat{D}\,\frac{AN}{DN}$$

The distance from the intersection point to the center of the end cap is then compared with the radius of the end cap to determine if the intersection point lies within the finite limits of the end cap.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
A	Radius of elliptical cylinder along x axis of the cylinder in wavelengths
AE	Distance from center of end cap to edge along line in x-z plane
AN	Dot product of vector from end cap to source and end cap unit normal
B	Radius of elliptical cylinder along y axis of the cylinder in wavelengths
CNC	The cosine of the angle between the z axis and the plane of end cap MC (angle measured in x-y plane)
CVE	Cosine of VE
D	Propagation direction in RCS
DHIT	Distance from source to nearest hit point
DHT	Distance from source to hit point

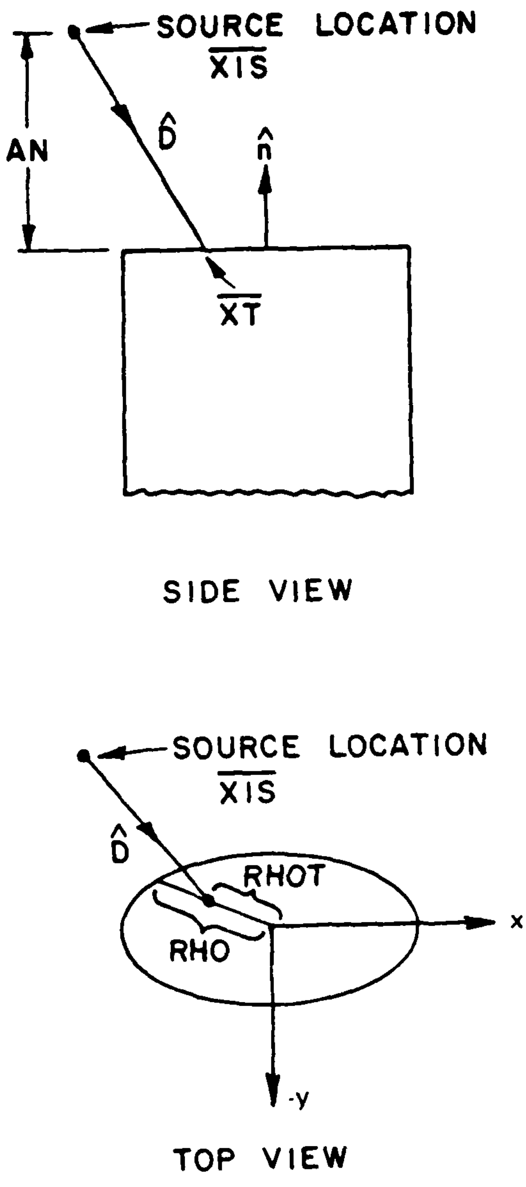


Figure 1. Geometry Used in CAPINT of a Ray Which Hits an End Cap

DN	Dot product of end cap unit normal and the ray scatter direction
LHIT	Set true if ray hits end cap
MC	End cap index variable
MD	Indicates which end caps are to be checked
NC	Sign change variable
RHO	Distance from z axis to point where [illegible] connecting the hit point and the origin hits the cylinder (2-D)
RHOT	Distance from z axis to point XT
SNC	The sine of the angle between the z axis and the plane of end cap MC (angle measured in x-y plane)
SVE	Sine of VE
VE	Elliptical angle defining hit point
XIS	X, Y, Z components of source (or ray origin) location in RCS. If MD > 0, variable becomes the location of hit point in RCS
XT	X, Y, Z components of point where ray hits end cap plane
ZC	Point where end cap intersects z axis of RCS. ZC(1) refers to the more positive end cap and ZC(2) refers to the more negative end cap.

5. I/O VARIABLES:

A. INPUT	LOCATION
A	/GEOMEL/
B	/GEOMEL/
CNC	/GEOMEL/
D	F.P.

	MD	F.P.
	SNC	/GEOMEL/
	XIS	F.P.
	ZC	/GEOMEL/
B.	OUTPUT	LOCATION
	DHIT	F.P.
	LHIT	F.P.
	XIS	F.P.

6. CALLING ROUTINES:

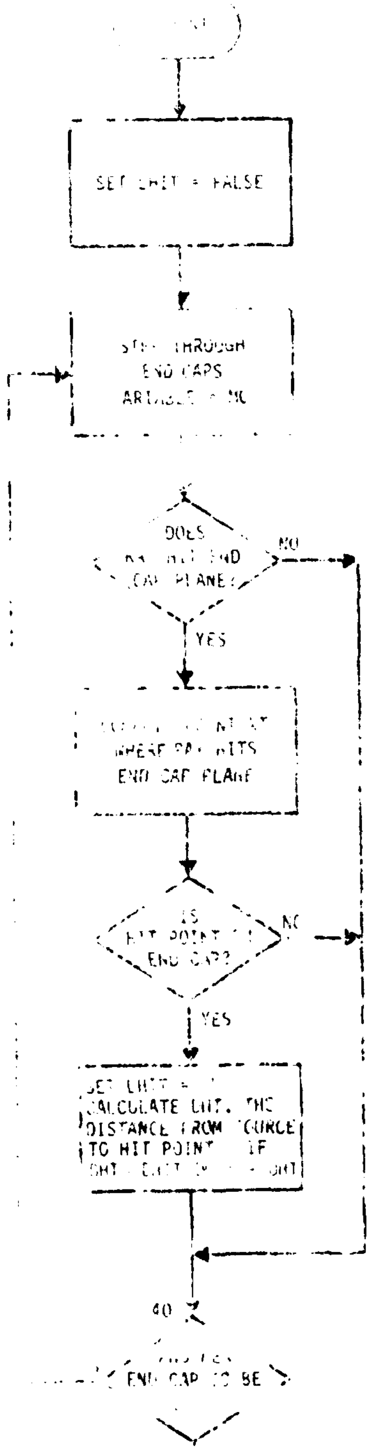
   CYLINT

   GEOMC

   REFCAP

7. CALLED ROUTINE:

   BTAN2


SET CHIT = FALSE
END CAPS
DOES
CAP PLANE?
NO
YES
WHERE RAY HITS
END CAP PLANE
IS
END CAP?
NO
YES
CALCULATE
DISTANCE FROM
TO HIT POINT
40
END CAP TO BE

1. NAME: CLSFIL (GTD, INPUT, MOM, OUTPUT)

2. PURPOSE: To close peripheral logical files.

3. METHOD: A FORTRAN end of file is written on the file and the file is rewound. Internal file flags are set.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
IFILE	Logical unit number.

5. I/O VARIABLES:

A. INPUT — LOCATION

INPUT	LOCATION
IFILE	F.P.

B. OUTPUT — LOCATION

OUTPUT	LOCATION
IOFILE	/IOFLES/
NDFILE	/IOFLES/

6. CALLING ROUTINES:*

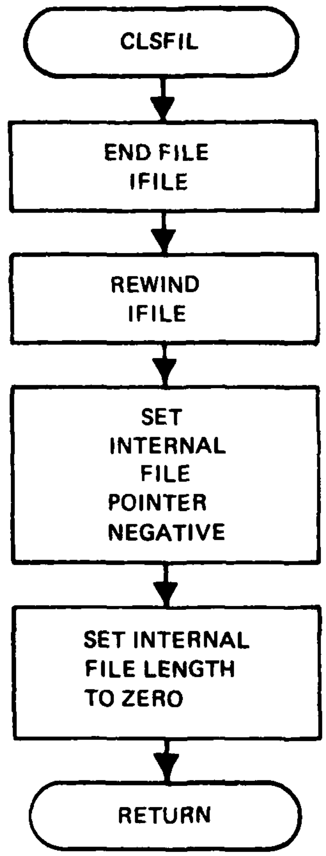
BUBBLE (1)	RWFILS (1,2,3,4)
CABC (3)	SOLDRV (3)
DECOMP (3)	STATFN (1,2,3,4)
DMPDRV (1,2,3,4)	SUBPAT (1)
ERROR (1,2,3,4)	SYMDEF (1,2,3,4)
FLDDRV (3)	TSKXQT (3)
GEODRV (1)	WRTCHK (1,2,3,4)
OPNFIL (1,2,3,4)	ZIJDRV (3)
PUTSYM (1,2,3,4)	

7. CALLED ROUTINES:

NONE

*1 - INPUT
2 - GTD
3 - MOM
4 - OUTPUT

CLSFIL (GTD, INPUT, MOM, OUTPUT)



1. NAME: CNTGND (MOM)

2. PURPOSE: Determine ground plane connections and their validity.

3. METHOD: For each of the MOM wire segments a check is made to determine whether or not one end or the other is connected to the ground plane in the x-y plane, if one has been specified. If an end is found to be in the ground plane, then its connectivity data are examined to see if there is another wire segment attached at this point. If there is, then an error condition is set and execution will terminate. If a valid attachment to ground is found, then the connectivity data are updated by showing that segment is connected to itself through the ground plane. This is an artifice to maintain the continuity of current constraint at the segment/ground interface.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
HAFLGH	Half the segment length
IBLK	Current block of geometry data
ICON	Connection data for segment
INEG	Connection data for negative end
IPOS	Connection data for positive end
LOCSAV	Logical file designation for last segment data set
LOCYRS	Pointer to segment data set entry in NDATBL array
NOGCFG	Flag set when multiple junction on ground plane detected
TOTCON	Number of wires connected to ground
ZC	Z coordinate of the segment center
ZN	Z coordinate minus half the segment length
ZP	Z coordinate plus half the segment length

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
IPERF	/AMPZIJ/
IP217	/GEODAT/
ISGTBL	/SEGMNT/
ISOFF	/ADEBUG/
ISON	/ADEBUG/
KOLNAM	/PARTAB/
LOCYRS	F.P.
LUPRNT	/ADEBUG/
MAXBLK	/SEGMNT/
MAXSEG	/SEGMNT/
NDATBL	/PARTAB/
NWIRE	/SEGMNT/
SEGTBL	/SEGMNT/
ZERO	/ADEBUG/

B. OUTPUT

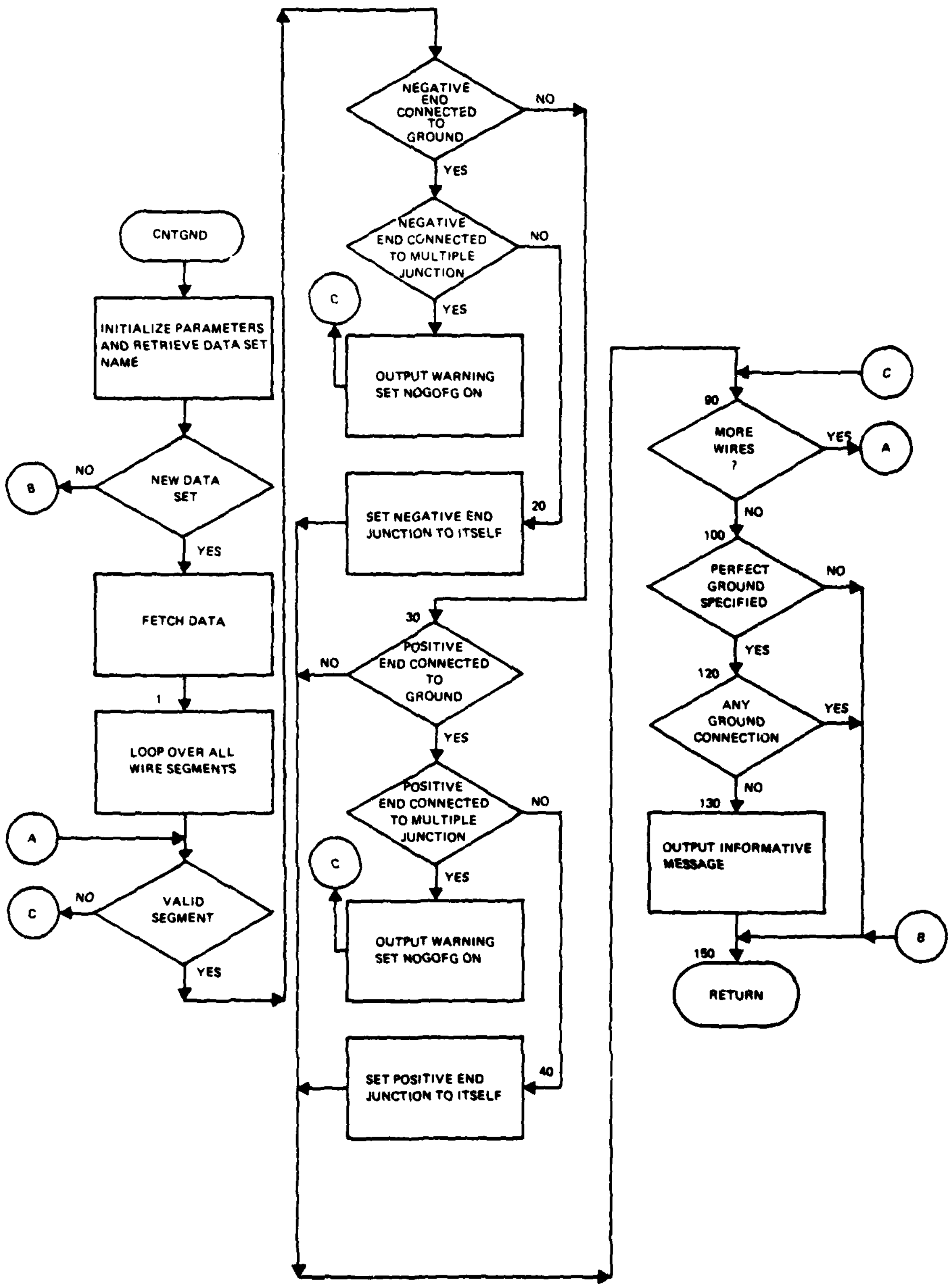
OUTPUT	LOCATION
IERRF	/ADEBUG/
ISGTBL	/SEGMNT/

6. CALLING ROUTINE: ZIJDRV

7. CALLED ROUTINES:

ASSIGN	STATIN
ERROR	STATOT
GETSEG	WLKBCK
GETSYM	

CNTGND (MOM)


CNTGND
INITIALIZE PARAMETERS AND RETRIEVE DATA SET NAME
NEW DATA SET
NO
YES
B
FETCH DATA
1
LOOP OVER ALL WIRE SEGMENTS
A
VALID SEGMENT
C
NEGATIVE END CONNECTED TO GROUND
NEGATIVE END CONNECTED TO MULTIPLE JUNCTION
OUTPUT WARNING SET NOGOFG ON
SET NEGATIVE END JUNCTION TO ITSELF
20
30
POSITIVE END CONNECTED TO GROUND
POSITIVE END CONNECTED TO MULTIPLE JUNCTION
OUTPUT WARNING SET NOGOFG ON
SET POSITIVE END JUNCTION TO ITSELF
40
90
MORE WIRES ?
100
PERFECT GROUND SPECIFIED
120
ANY GROUND CONNECTION
130
OUTPUT INFORMATIVE MESSAGE
150
RETURN

1. NAME: CNVGTD (INPUT)

2. PURPOSE: Detect and link all wires which are connected to plates. Update the SEGTBL entries for connected wires.

3. METHOD: Both ends of every wire segment in SEGTBL are checked to see if an end lay within the domain of a GTD plate. Subroutine PLTSEG is called with an endpoint and returns the plate number to which the end is attached. The segment may be attached to any point on the plate, and more than one segment may be connected to the same plate. If the end attaches to no plate, zero is returned.

When a connection is detected, the SEGTBL entry for that segment is updated by placing the segment number in the left half (negative end connected) or right half (positive end connected) of the connection data word SEGTBL (9,I). These connections will be treated like wires connected to ground in SEJCON and ZIJSET so that the segment's basis function may be properly distributed using image theory.

An error will occur if a wire segment is attached to something else (another wire, a patch, or ground) at the same end that is attached to a plate. Thus, no multiple junctions are allowed at plate attachment points. Moreover, for good current modeling, only one end of any segment should be connected to a patch, plate, or ground.

4. INTERNAL VARIABLES:

VARIABLE	DESCRIPTION
DX,DY,DZ	Direction cosines of wire segment vector orientation
FLEN	Segment length
I	Wire segment loop index
ICON	Segment connection data word
IEND1	Negative endpoint number
IEND2	Positive endpoint number
IHIT	Plate number to which segment is attached (zero if no attachment)
ILIM	Number of segments in this SEGTBL block
INEG	Connection data for negative endpoint

IPOS	Connection data for positive endpoint
IS	Wire segment number
IWRBLK	Number of SEGTBL blocks containing wire segments
K	SEGTBL block loop index
NCON	Total number of segment-plate connections
NOGOFG	Internal error flag
XC,YC,ZC	Coordinates of segment center
XN,YN,ZN	Coordinates of segment negative endpoint
XP,YP,ZP	Coordinates of segment positive endpoint

5. I/O VARIABLES:

A. INPUT — LOCATION

INPUT	LOCATION
IP217	/GEODAT/
ISGTBL	/SEGMNT/
ISOFF	/ADEBUG/
ISON	/ADEBUG/
MAXBLK	/SEGMNT/
MAXSEG	/SEGMNT/
NWIRE	/SEGMNT/
SEGTBL	/SEGMNT/
UPDBLK	/SEGMNT/

B. OUTPUT — LOCATION

OUTPUT	LOCATION
IERRF	/ADEBUG/
ISGTBL	/SEGMNT/
UPDBLK	/SEGMNT/

6. CALLING ROUTINE: LNKGTD

7. CALLED ROUTINES:

ASSIGN

ERROR

GETSEG

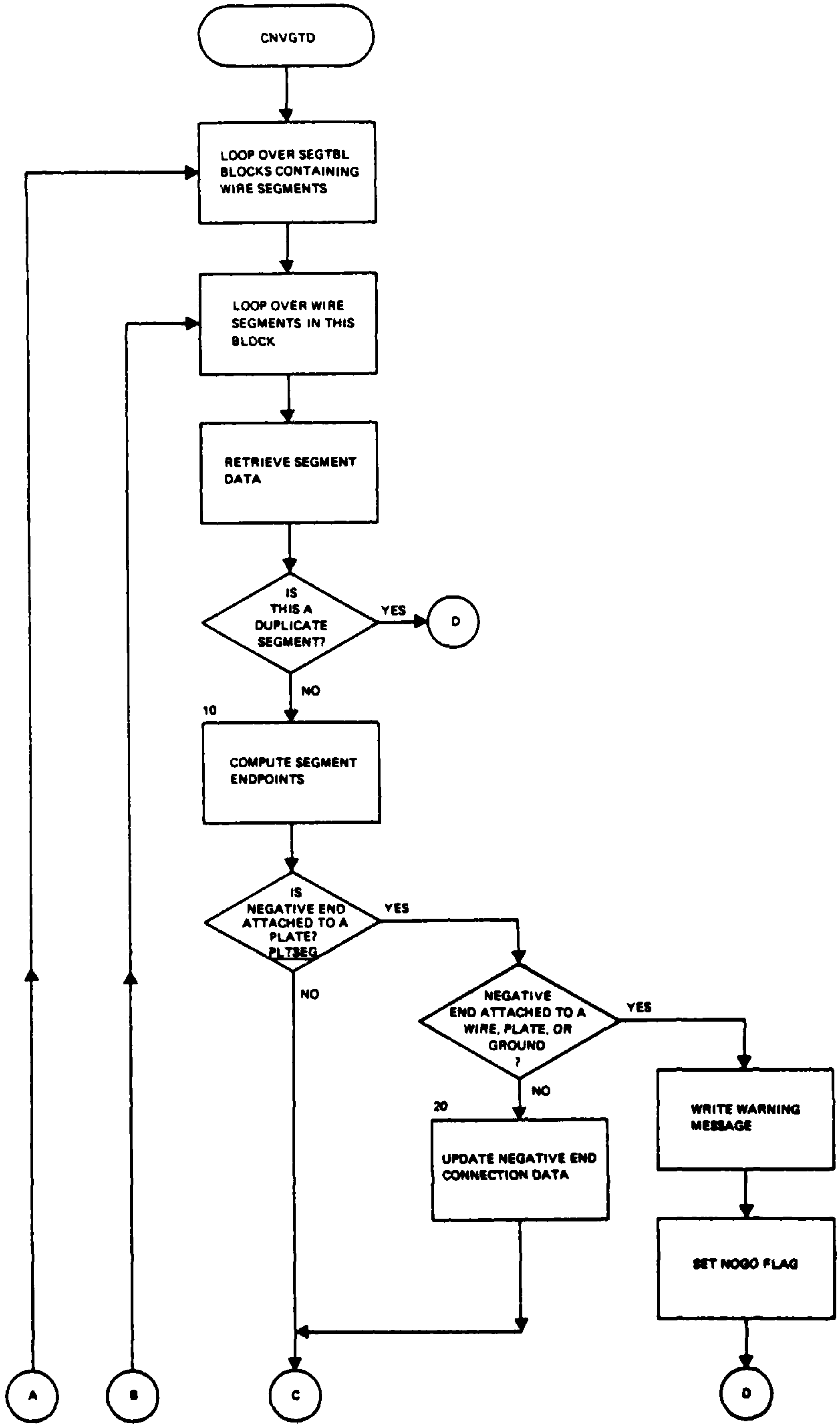
PLTSEG

STATIN

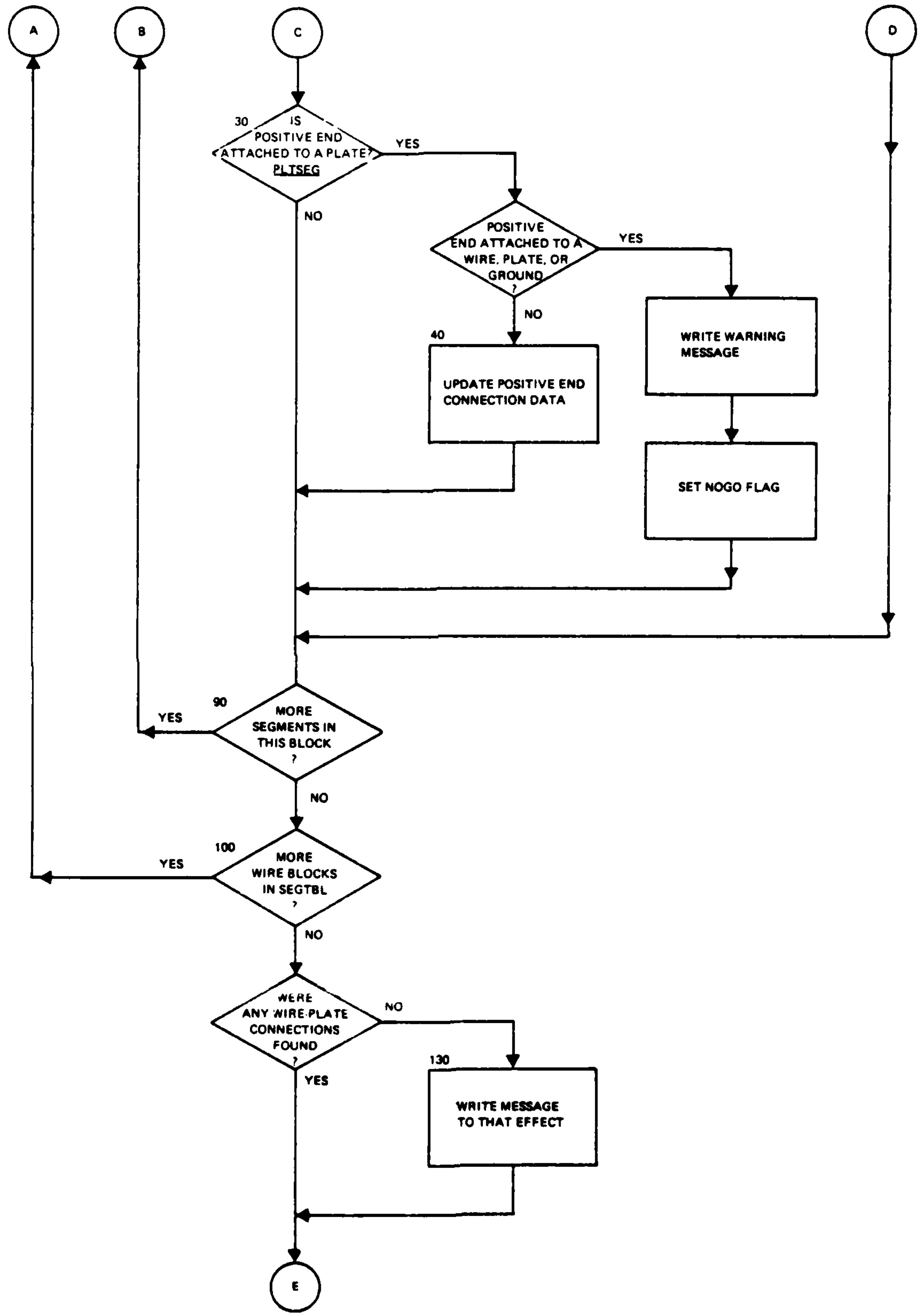
STATOT

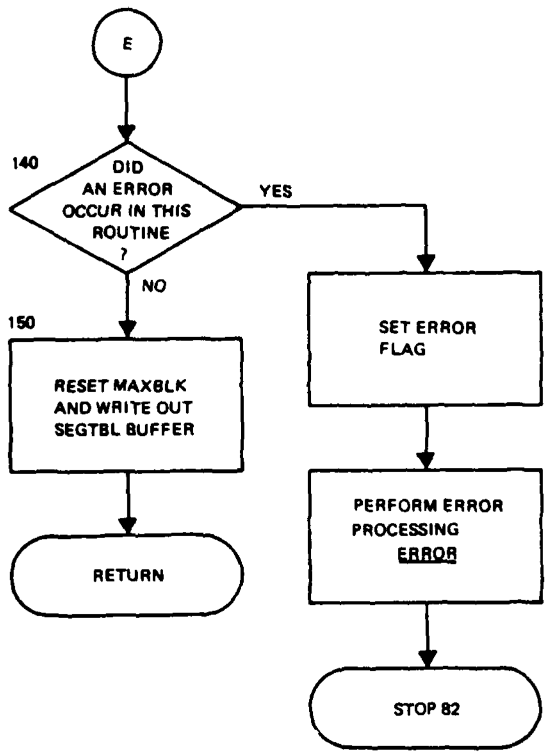
WLKBCK

CNVGTD (INPUT)



CNVGTD (INPUT)




E
140
DID AN ERROR OCCUR IN THIS ROUTINE ?
YES
NO
150
RESET MAXBLK AND WRITE OUT SEGTBL BUFFER
RETURN
SET ERROR FLAG
PERFORM ERROR PROCESSING
ERROR
STOP 82

1. NAME: CNVTST (GTD, MOM)

2. PURPOSE: Convergence test for Rombert integration routine ROMBNT.

3. METHOD: If value of F2 is less than or equal to 10^{-38} set CNVTST = 0, otherwise set to $|(F1 - F2)/F2|$.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
F1	First trapezoidal rule result
F2	Second trapezoidal rule result

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
F1	F.P.
F2	F.P.

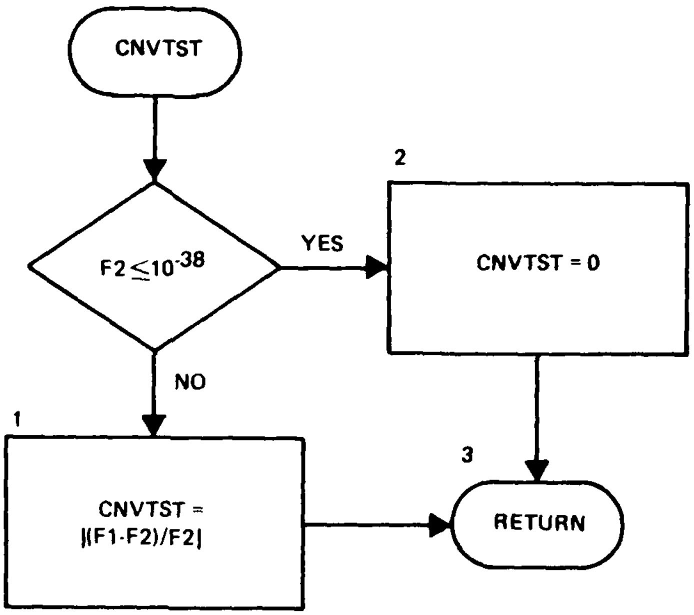
B. OUTPUT

OUTPUT	LOCATION
CNVTST	FUNCTION

6. CALLING ROUTINE: ROMBNT

7. CALLED ROUTINE: None.

CNVTST (GTD, MOM)



1. NAME: CONJUG (MOM)

2. PURPOSE: This subroutine conjugates a complex matrix.

3. METHOD: For a complex matrix every imaginary element is replaced by its negative.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
L	Total number of elements in the array
NC	Number of columns in the complex matrix
NR	Number of rows in the complex matrix
Z	The complex matrix

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
NC	F.P.
NR	F.P.
Z	F.P.

B. OUTPUT

OUTPUT	LOCATION
Z	F.P.

6. CALLING ROUTINE:

SOLDRV

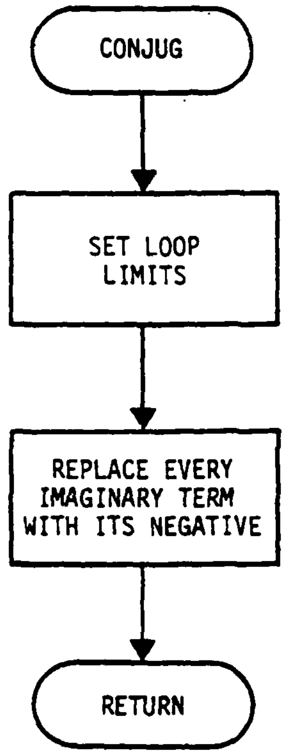
7. CALLED ROUTINES:

ASSIGN

STATIN

STATOT

WLKBCK


CONJUG
SET LOOP LIMITS
REPLACE EVERY IMAGINARY TERM WITH ITS NEGATIVE
RETURN

1. NAME: CONVRT (GTD, INPUT, MOM, OUTPUT)

2. PURPOSE: Decodes internal symbolic codes to external, left justified blank filled BCD.

3. METHOD: By dividing the input by 2 raised to the power of the number of bits per byte, the encoded characters are loaded into an array. By comparing this array with the internal BCD representation, the output is constructed.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
ICHAR	BCD representation loaded in proper position
ID	Output word
IND	Index to INTBCD array
INTBCD	Integer array corresponding to BCD representation of characters (machine dependent)
INTWRD	Intermediate output word
NBYTES	Number of bits per word
NBYTSZ	Number of bits per byte
NSH	Power of 2 such that multiplication by this number results in a bit pattern being shifted by one byte.
NW	Input NWORD shifted to right by one byte
NWORD	Internal NW
NXTWRD	Word NWORD shifted to right by one byte

5. I/O VARIABLES:

A. INPUT — LOCATION

INPUT	LOCATION
NW	F.P.

B. OUTPUT — LOCATION

OUTPUT	LOCATION
ID	F.P.

6. CALLING ROUTINES:*

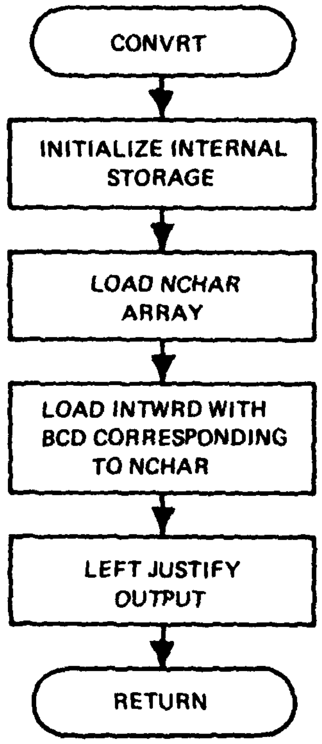
BACSUB (3)	POSTIP (1,2)
BANDIT (3)	POSTPR (1)
DMPDRV (1,2,3,4)	PREPAR (1)
EGFMAT (3)	PRTKJ (2,3)
EXCDRV (2,3)	PRTSYM (3)
FLDDRV (2,3,4)	PUTKWV (1,2,3,4)
FLDOUT (4)	PUTSYM (1,2,3,4)
FNDREC (1,2,3,4)	REBLCK (3)
GEMACS (1)	RESTRT (1)
GEODRV (1)	RWFILS (1,2,3,4)
GETARG (1,2,3,4)	SETDRV (3)
GETGEO (2,3,4)	SOLDRV (3)
GETKWV (1,2,3,4)	SYMDEF (1,2,3,4)
GETSYM (1,2,3,4)	SYMUPD (1,2,3,4)
LODDRV (3)	TSKXQT (1,2,3,4)
LUDDRV (3)	ZIJDRV (2,3)
PLTDRV (1)	ZIJSET (3)

7. CALLED ROUTINES:

NONE

*1 - INPUT
2 - GTD
3 - MOM
4 - OUTPUT

CONVRT (GTD, INPUT, MOM, OUTPUT)



1. NAME: COORDS (INPUT)

2. PURPOSE: Performs all coordinate transformations and rotations for the GEMACS computer code.

3. METHOD: The angles of rotation and the translation vectors are read either from the input argument list or are stored in common. The angles used are the polar angles as defined in the User Manual. The translation and rotation take place by a call to subroutines TRNLAT and ROTATE, respectively.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
ATTACH	Logical TRUE for the ATTACH operation
CX	The x component of the specified coordinate system
CY	The y component of the specified coordinate system
CZ	The z component of the specified coordinate system
DX,DY,DZ	Negative of CX,CY,CZ, respectively
IC	Coordinate system index
ICS	Absolute value of IC
ICSAV	Saved coordinate system index
IDCSYS	Coordinate system identification number
IERRF	Error flag
II	Coordinate system index to IDCSYS
LSTCSY	Last value for a coordinate system
ROX	Rotation about x axis
ROY	Rotation about y axis
ROZ	Rotation about z axis
RX,RY,RZ	Rotation angles for coordinate system II

X Input/output x variable that is to be transformed

Y Input/output y variable that is to be transformed

Z Input/output z variable that is to be transformed

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
ATTACH	F.P.
CX	/CSYSTM/
CY	/CSYSTM/
CZ	/CSYSTM/
DBGPRT	/ADEBUG/
IC	F.P.
IDCSYS	/CSYSTM/
LSTCSY	/CSYSTM/
N	F.P.
ROX	/CSYSTM/
ROY	/CSYSTM/
ROZ	/CSYSTM/
X	F.P.
Y	F.P.
Z	F.P.

B. OUTPUT

OUTPUT	LOCATION
IERRF	/ADEBUG/
X	F.P.

Y	F.P.
Z	F.P.

6. CALLING ROUTINES:

   PATCH

   WYRDRV

7. CALLED ROUTINES:

   ASSIGN

   ERROR

   ROTATE

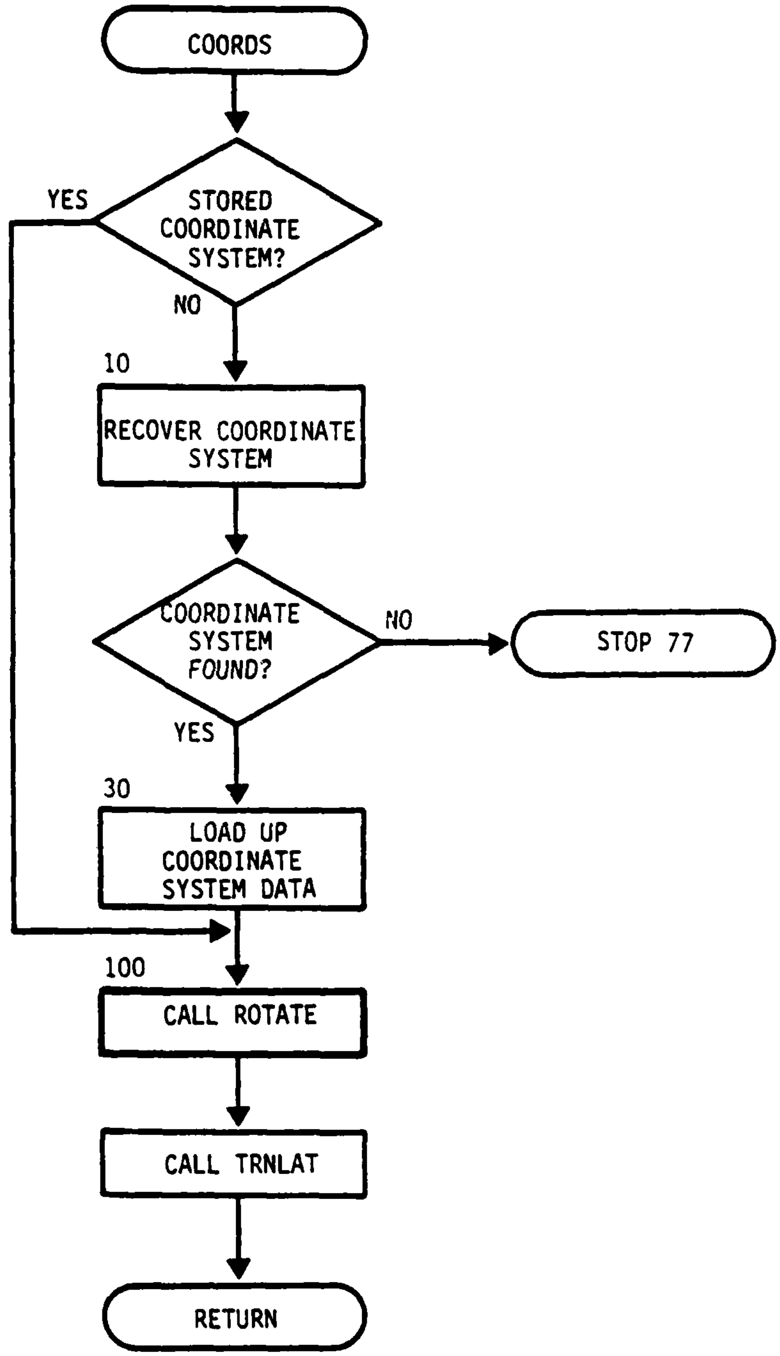
   STATIN

   STATOT

   TRNLAT

   WLKBCK

COORDS (INPUT)



1. NAME: CYAXIS (GTD)

2. PURPOSE: To determine the conversion constants required to rotate the geometry from the global GEMACS coordinate system (SEGTBL entries) to the cylinder-centered reference coordinate system (RCS).

3. METHOD: The variable ICSYS represents the coordinate system in which the cylinder geometry was input. It is used to define the reference coordinate system (RCS) for the GTD calculations. If a cylinder is present in the geometry, the RCS origin is defined in the center of the cylinder. The z axis runs parallel to the cylinder walls and out the more positive end cap. The major radius of the elliptic cylinder is defined on the x axis. The minor radius of the cylinder is defined on the y axis. A figure of this coordinate system is shown in figure 1.

   This subroutine calls subroutine ROTATE to determine how the RCS axes are represented in x, y, z components of the global coordinate system. The x, y, z components defining each axis will be used to rotate all the other geometry locations to the RCS. An illustration of this is shown in figure 2. The translation of the geometry locations is performed in the calling routine GTDDRV.

   If a cylinder is not present in the geometry, the RCS is defined as the global coordinate system and CYAXIS is not called.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
ICSYS	Number of the coordinate system to be used as the reference coordinate system.
IDCSYS	Contains coordinate system identification number.
LSTCSY	Value of the last coordinate system used for geometry input.
LUPRNT	Output file.
RX	Rotation angle in degrees about the x axis of the global system.
RY	Rotation angle in degrees about the y axis of the global system.
RZ	Rotation angle in degrees about the z axis of the global system.

CYAXIS (GTD)

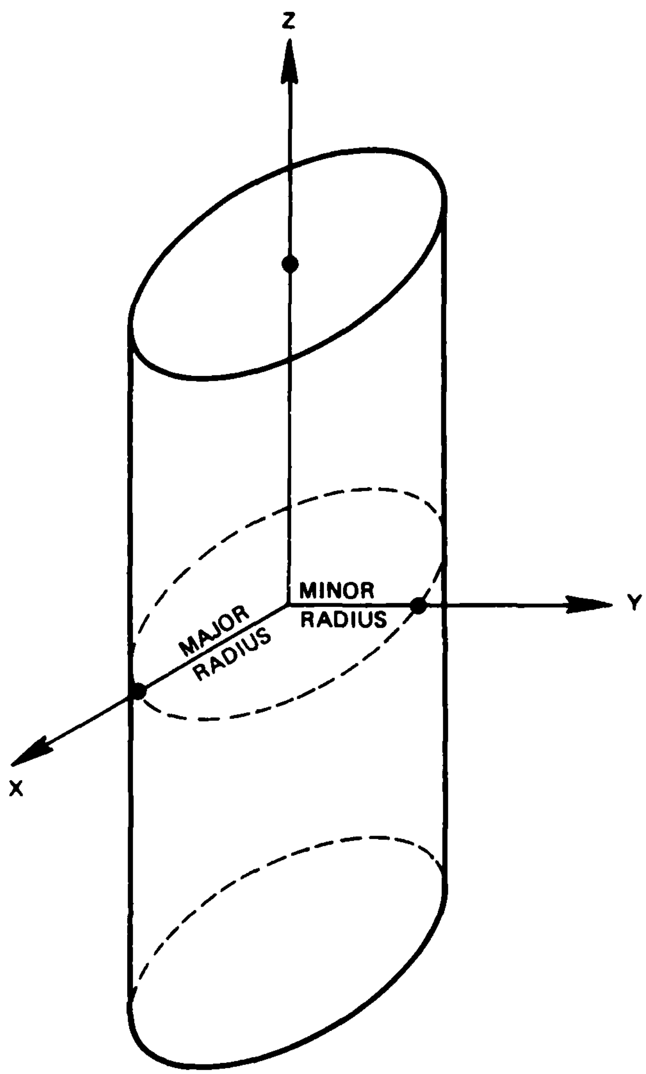


Figure 1. Cylinder Centered Coordinate System Used as the Reference Coordinate System for the GTD Calculations

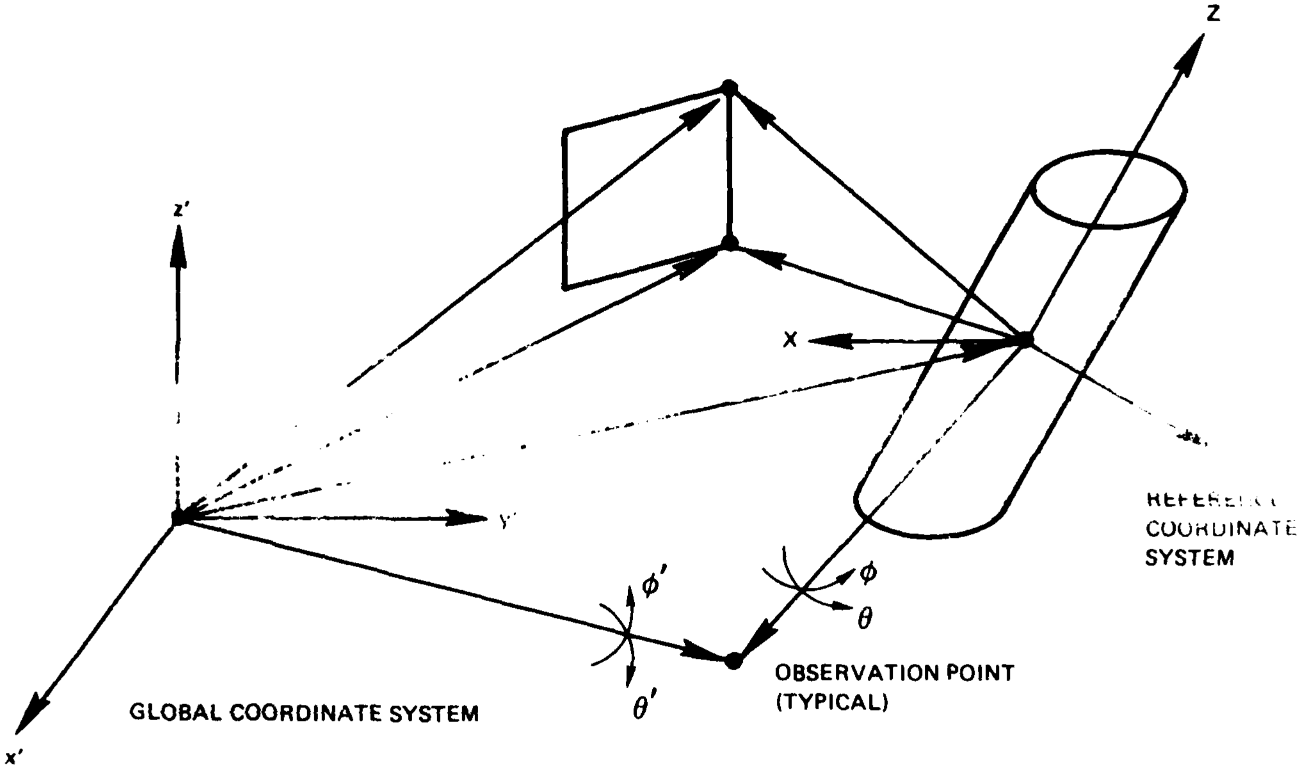


Figure 2. Illustration Showing the Reference Coordinate System and the Global Coordinate System

XCL	Direction cosine of the reference coordinate system x axis unit vector in global coordinate system components.
YCL	Direction cosines of the reference coordinate system y axis unit vector in global coordinate system components.
ZCL	Direction cosine of the reference coordinate system z axis unit vector in the global coordinate system components.

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
ICSYS	F.P.
IDCSYS	/CSYSTM/
LSTCSY	/CSYSTM/
LUPRNT	/ADEBUG/
ROX	/CSYSTM/
ROY	/CSYSTM/
ROZ	/CSYSTM/
RX	/CSYSTM/
RY	/CSYSTM/
RZ	/CSYSTM/

B. OUTPUT

OUTPUT	LOCATION
ICSYS	F.P.
XCL	F.P.
YCL	F.P.
ZCL	F.P.

6. CALLING ROUTINE:

GTDDRV

7. CALLED ROUTINES:

ASSIGN

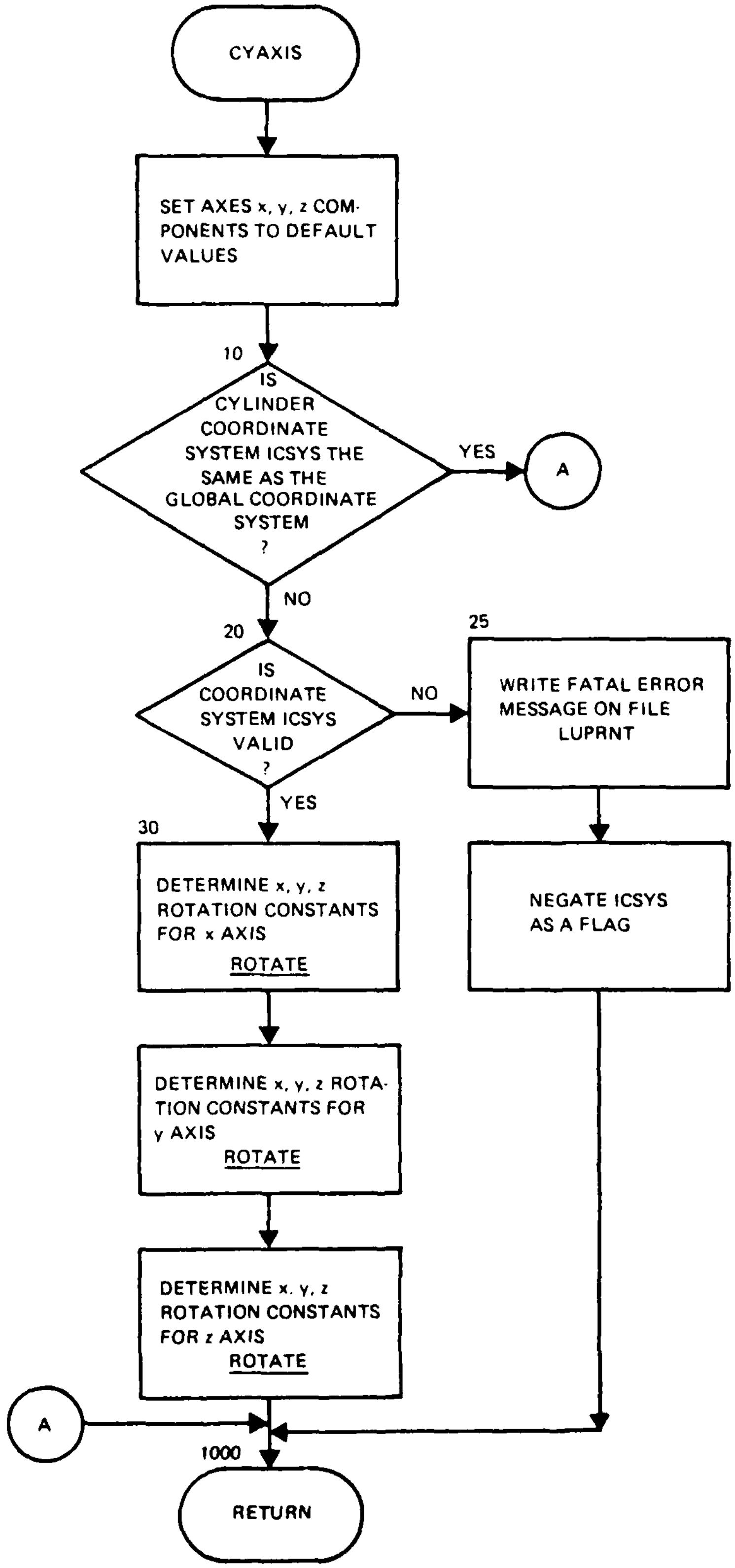
ROTATE

STATIN

STATOT

WLKBCK

CYAXIS (GTD)


CYAXIS
SET AXES x, y, z COMPONENTS TO DEFAULT VALUES
10
IS CYLINDER COORDINATE SYSTEM ICSYS THE SAME AS THE GLOBAL COORDINATE SYSTEM ?
YES
A
NO
20
IS COORDINATE SYSTEM ICSYS VALID ?
NO
25
WRITE FATAL ERROR MESSAGE ON FILE LUPRNT
NEGATE ICSYS AS A FLAG
YES
30
DETERMINE x, y, z ROTATION CONSTANTS FOR x AXIS
ROTATE
DETERMINE x, y, z ROTATION CONSTANTS FOR y AXIS
ROTATE
DETERMINE x, y, z ROTATION CONSTANTS FOR z AXIS
ROTATE
A
1000
RETURN

1. NAME: CYLINT (GTD)

2. PURPOSE: To determine if a ray traveling from a given direction will intersect the elliptic cylinder.

3. METHOD: First the distance from the source to the cylinder axis is compared to the radius of the cylinder to see if the source can illuminate the curved surface of the cylinder. If it cannot, as illustrated in figure 1, a check is made to see if the source is inside the cylinder. If it is not inside, subroutine CAPINT is called to determine if the ray hits an end cap.

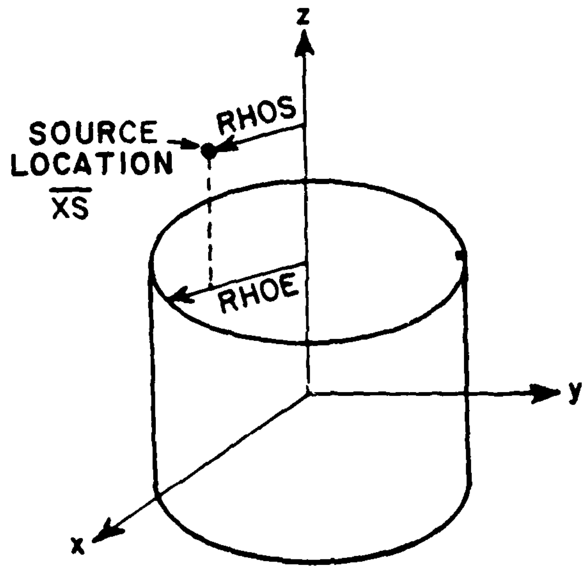
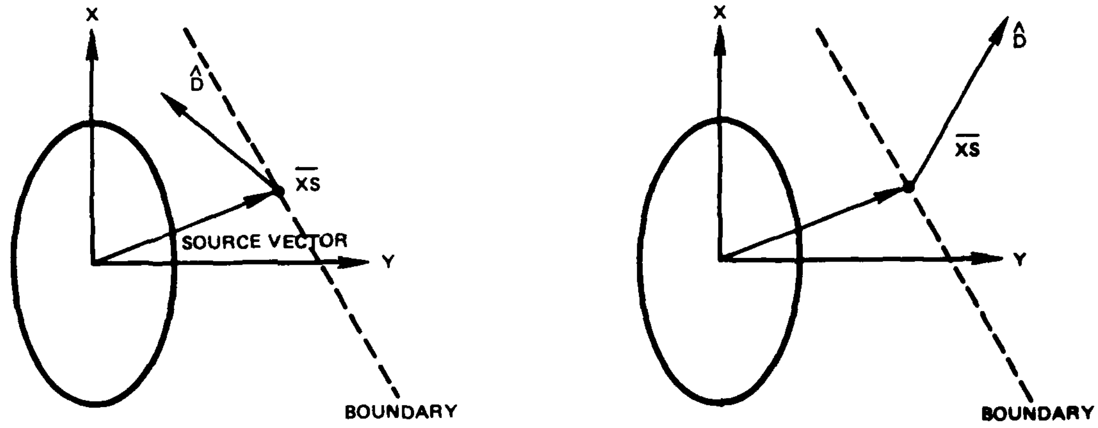


Figure 1. *Illustration of Source Which Cannot Illuminate* Curved Cylinder Surface. RHOS $\leq$ RHOE

If it is possible to illuminate the curved surface (RHOS > RHOE), a check is made to determine if the ray is aimed in the direction of the cylinder or not. If the dot product of the propagation direction in the x-y plane and the source location vector in the x-y plane is less than or equal to zero, the ray travels towards the cylinder. Figure 2 shows the two possibilities.

The next step is to determine if a ray traveling towards the cylinder intersects the cylinder x-y plane cross section.



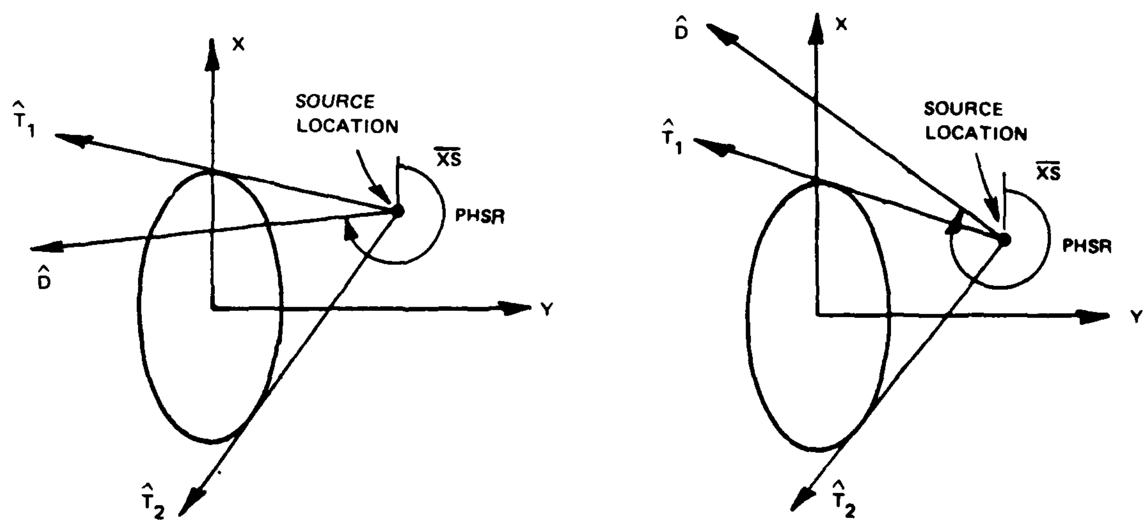
a) Ray Travels Towards Cylinder

b) Ray Does Not Travel Towards Cylinder

Figure 2. Illustration of Possible Ray Paths

If $\hat{D} \cdot \hat{T}_1 \geq \hat{T}_1 \cdot \hat{T}_2$ and $\hat{D} \cdot \hat{T}_2 \geq \hat{T}_1 \cdot \hat{T}_2$, the ray hits the cylinder cross section (see figure 3a).

If either $\hat{D} \cdot \hat{T}_1 < \hat{T}_1 \cdot \hat{T}_2$ or $\hat{D} \cdot \hat{T}_2 < \hat{T}_1 \cdot \hat{T}_2$, the ray does not hit the cylinder cross section (see figure 3b). This case is then dismissed.



a) Ray Hits the Cylinder Cross Section

b) Ray Does Not Hit the Cylinder Cross Section

Figure 3. Illustrations of Rays Traveling Towards Cylinder

If the cylinder cross section is hit, the subroutine then solves a quadratic equation to determine the two intersection points on the cylinder cross section. (As shown in figure 3a, the propagating ray pierces the cylinder in two places.) The details of this technique are given on pages 90-96 of reference A. The closer of the two intersections to the source is selected to obtain the x and y coordinates of the hit point.

The z-location of the hit point is then determined and checked to see if it lies on the curved surface within the cylinder end caps, as shown in figure 4. If it does not, the ray may hit a cylinder end cap so CAPINT is called. If it does hit the cylinder, the distance from the source to the hit point is calculated and the subroutine task is complete.

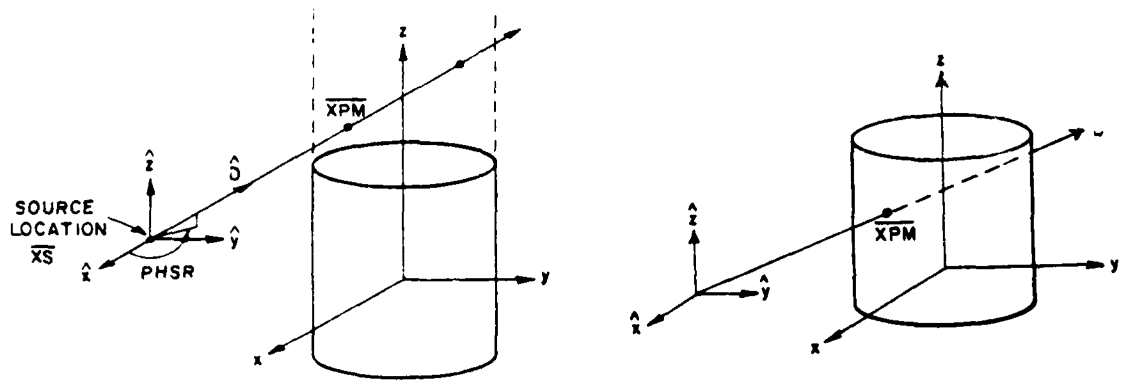


Figure 4. Illustration of Rays that: a) Hit Cylinder Projection but not Cylinder and b) Hit Cylinder

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
A	Radius of elliptical cylinder along the x-axis of the cylinder in wavelengths
B	Radius of elliptical cylinder along the y-axis of the cylinder in wavelengths

BM	Parameter used in computing hit point 1
BPL	Parameter used in computing hit point 2
BTD	x and y components of unit vectors of rays from ray origin tangent to cylinder
BTS	Defines unit vectors of the two source rays tangent to the cylinder, where the unit vector for the source ray tangent to tangent point 1 is given by (figure 3): $T1 = BTS(1)\hat{x} + BTS(2)\hat{y}$ and the unit vector for the source ray tangent to tangent point 2 is given by: $T2 = BTS(3)\hat{x} + BTS(4)\hat{y}$
CPS	Cosine of PHSR
CTC	Cotangent of the angle between the z-axis and the plane of end cap MC (measured in x-z plane)
CVE	Cosine of VE
D	Ray propagation direction in RCS
D12	Dot product of source vectors tangent to the cylinder (in x-y plane)
DD1	Dot product of the propagation direction and T1 tangent unit vector
DD2	Dot product of the propagation direction and T2 tangent unit vector
DHIT	Distance from source to (nearer) hit point
DM	Distance from source to hit point 1
DPL	Distance from source to hit point 2
DTD	Dot product of ray origin vectors tangent to the cylinder (x-y plane)
DTS	The dot product of the two source vectors tangent to the cylinder: $DTS = T1 \cdot T2$

DXY	Dot product of ray from origin to source and propagation direction (in x-y plane)
LBDF	Set true if ray origin XS is not the source location
LCYL	Set true if a cylinder is present
LHIT	Set true if ray hits cylinder or end cap
PHSR	Phi angle of propagation direction in RCS
RHOE	Radius from z-axis to point where ray from origin to source intersects the cylinder
RHOS	Distance from source to z-axis
SPS	Sine of PHSR
SVE	Sine of VE
TX1	X component of tangent unit vector, T1
TX2	X component of tangent unit vector, T2
TY1	Y component of tangent unit vector, T1
TY2	Y component of tangent unit vector, T2
VE	Elliptical angle of source location in RCS x-y plane
VM	Elliptical angle defining hit point 1 on cylinder in RCS x-y plane
VPL	Elliptical angle defining hit point 2 on cylinder in RCS x-y plane
VT	Elliptical angle defining hit point on cylinder closer to source
VTD	Not used (from subroutine TANG)
XPM,YPM,ZPM	Used in several cases to define hit point (x, y, z components in RCS) on cylinder (Used in various forms)
XS	Source location (or point from which ray originates) in RCS

ZC	Point where end cap intersects z axis of RCS; ZC(1) refers to the more positive end cap and ZC(2) refers to the more negative

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
A	/GEOMEL/
B	/GEOMEL/
BTS	/BNDSCL/
CTC	/GEOMEL/
D	F.P.
DTS	/BNDSCL/
LBDF	F.P.
LCYL	/LPLCY/
PHSR	F.P.
XS	F.P.
ZC	/GEOMEL/

B. OUTPUT

OUTPUT	LOCATION
DHIT	F.P.
LHIT	F.P.

6. CALLING ROUTINES:

DIFPLT

DPLRCL

DPLRPL

GEOM

GTDDRV

INCFLD

RCLDPL

RCLRPL

REFPLA

RPLDPL

RPLRCL

RPLRPL

RPLSCL

SCLRPL

7. CALLED ROUTINES:

BTAN2

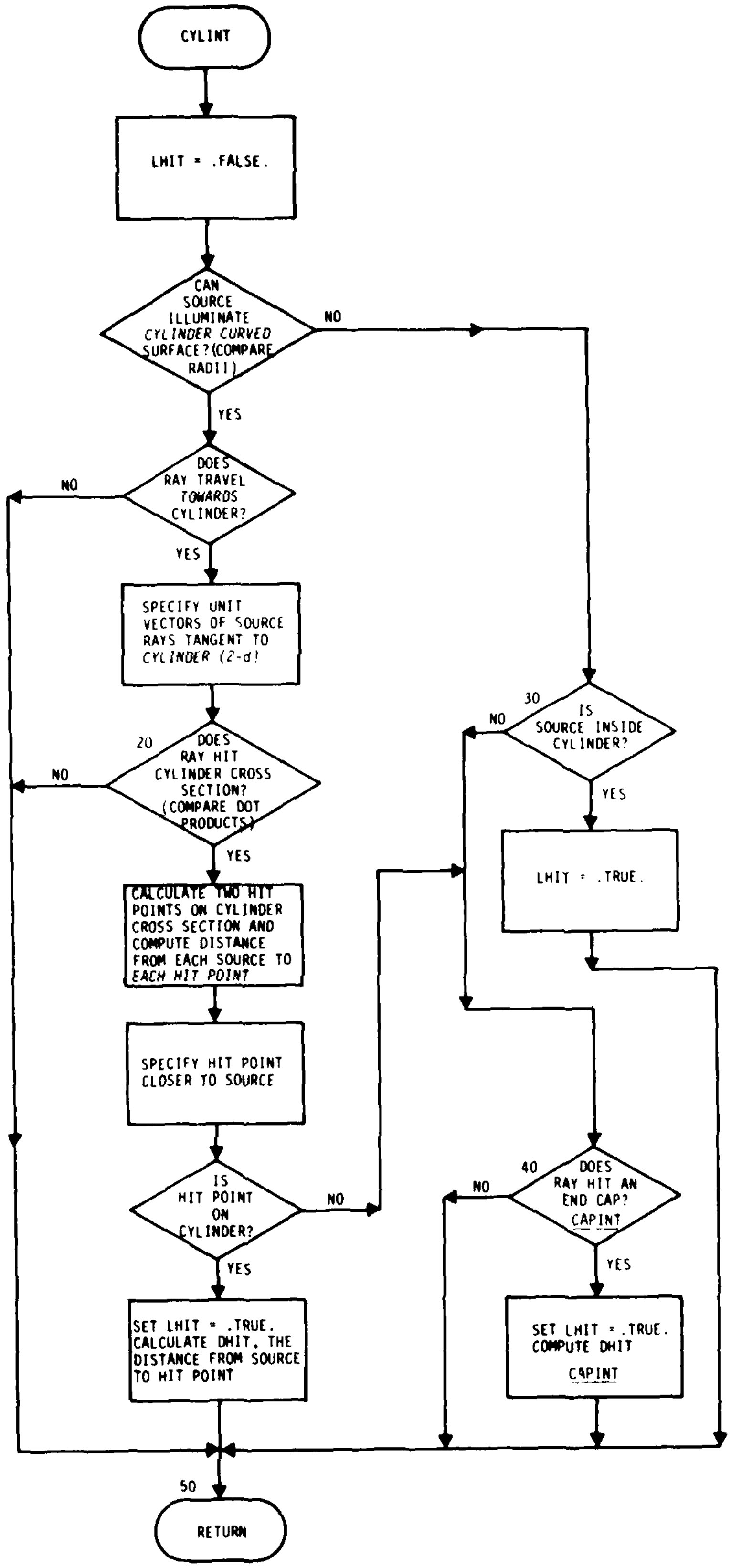
CAPINT

TANG

8. REFERENCES:

A. R. J. Marhefka, "Analysis of Aircraft Wing-Mounted Antenna Patterns," Report 2902-25, June 1976, The Ohio State University ElectroScience Laboratory, Department of Electrical Engineering; prepared under Grant No. NGL 36-008-138 for National Aeronautics and Space Administration.

CYLINT (GTD)


CYLINT
LHIT = .FALSE.
CAN SOURCE ILLUMINATE CYLINDER CURVED SURFACE? (COMPARE RADII)
NO
YES
DOES RAY TRAVEL TOWARDS CYLINDER?
NO
YES
SPECIFY UNIT VECTORS OF SOURCE RAYS TANGENT TO CYLINDER (2-d)
20
DOES RAY HIT CYLINDER CROSS SECTION? (COMPARE DOT PRODUCTS)
NO
YES
CALCULATE TWO HIT POINTS ON CYLINDER CROSS SECTION AND COMPUTE DISTANCE FROM EACH SOURCE TO EACH HIT POINT
SPECIFY HIT POINT CLOSER TO SOURCE
IS HIT POINT ON CYLINDER?
NO
YES
SET LHIT = .TRUE. CALCULATE DHIT, THE DISTANCE FROM SOURCE TO HIT POINT
30
IS SOURCE INSIDE CYLINDER?
NO
YES
LHIT = .TRUE.
40
DOES RAY HIT AN END CAP? CAPINT
NO
YES
SET LHIT = .TRUE. COMPUTE DHIT CAPINT
50
RETURN

1. NAME: CYLNDR (INPUT)

2. PURPOSE: To store raw GTD cylinder geometry data into the segment table and make an entry into the point table for later geometry linkage by LNKGTD.

3. METHOD: WYRDRV calls CYLNDR whenever a CY command is encountered in the geometry data. CYLNDR interprets the items as scanned by SCAN and extracts values for cylinder number, major and minor radii of cross section, length, and coordinate system. If an optional item is not present, the values from the last CY command are used. All optional values are initially set to zero.

The format of the CY command is

	CY	ncyl	[aa]	[bb]	[len]	[icsys]	
array indices	1	2	3	4	5	6	NVAL/VAL
			1	2	3	4	IO/FO
		NTINT	NTFLPT	NTFLPT	NTFLPT	NTINT	argument type

ncyl = user cylinder number

aa = x axis radius

bb = y axis radius

len = length

icsys = number of previously defined coordinate system to be used to translate cylinder center and rotate cylinder axes.

CYLNDR places the following values in SEGTBL and PNTTBL:

	SEGTBL		PNTTBL	
1	ITAG/IS		0	Point number id = ITAG/ncyl
2	0		icsys	
3	AA		FLEN	
4	BB			
5	FLEN			
6	XTRL	translation		
7	YTRL			
8	ZTRL			
9	THETA	rotation angles		
10	PHI			
11	cyl/icsys			

SEGTBL data are used in the GTD module. PNTTBL data are used later by LNKGTD in the INPUT module to link cylinders with end caps.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
AA	X-axis cylinder radius (m)
BB	Y-axis cylinder radius (m)
ERRFLG	Internal flag to indicate command error (integer)
FLEN	Cylinder length (m)
FØ	Array containing real values of arguments
ICSYS	Coordinate system number
IPT	Packed word containing tag and cylinder numbers. The tag and cylinder numbers each occupy 16 bits of the word.
ISG	Cylinder segment number (assigned by PUTSEG)
ITAG	Cylinder tag number
ITPARG	Array of variable types (real, integer) for argument field
IØ	Array containing integer values of arguments
MXCYAR	Maximum number of arguments allowed on cylinder command
NSGTBL	Number of SEGTBL entries required for cylinder geometry
NUMCY	The cylinder number assigned by the user
THETA,PHI	Angles defining cylinder axis rotation (radians)
XTRL,YTRL,ZTRL	Coordinates defining cylinder center translation (m)

5. I/O VARIABLES:

A. INPUT — LOCATION

INPUT	LOCATION
ICYTAG	/GTDDAT/
IP217	/GEODAT/
ISOFF	/ADEBUG/
ISON	/ADEBUG/
NARGS	/SCNPAR/
NCODE	/SCNPAR/
NTFLPT	/ADEBUG/
NTINT	/ADEBUG/
NVAL	/SCNPAR/
SCALE	/SEGMNT/
VAL	/SCNPAR/

B. OUTPUT — LOCATION

OUTPUT	LOCATION
NOGOFG	/SCNPAR/

6. CALLING ROUTINE: WYRDRV

7. CALLED ROUTINES:

ASSIGN

GTDCS

PUTPNT

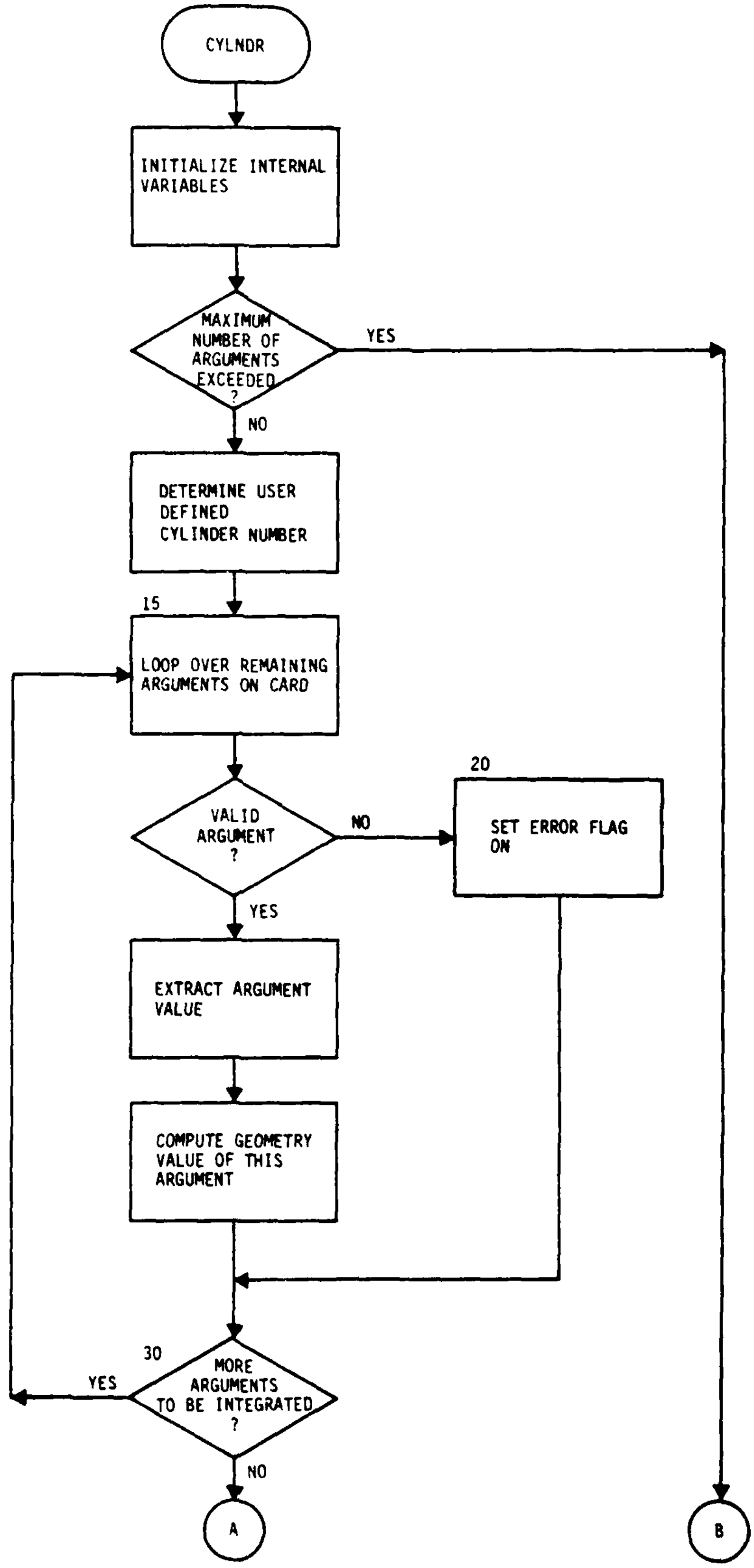
PUTSEG

STATIN

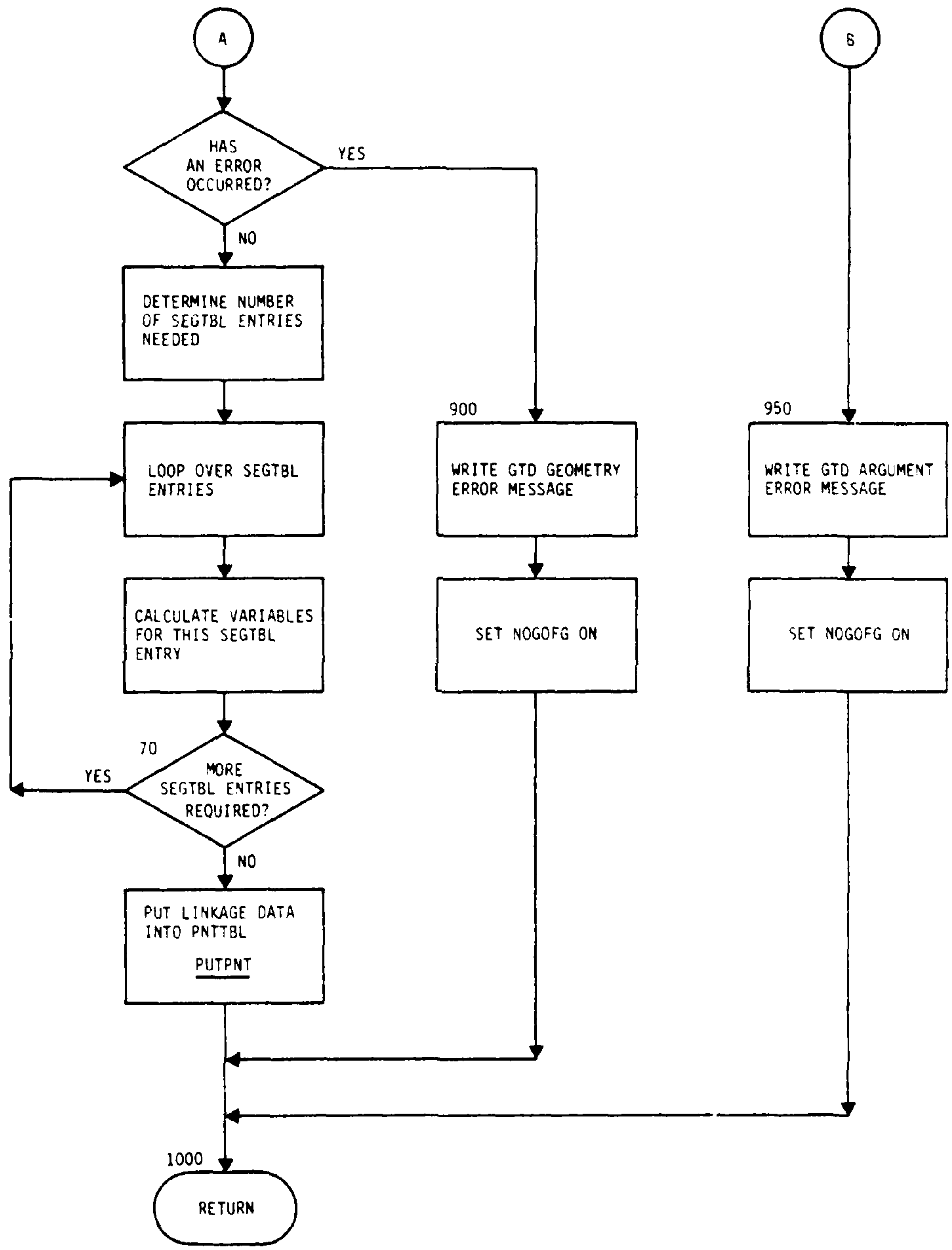
STATOT

WLKBCK

CYLNDR (INPUT)



CYLNDR (INPUT)



1. NAME: DECOMP (MOM)

2. PURPOSE: Subroutine DECOMP performs lower-upper triangular decomposition.

3. METHOD: The method employed to decompose matrices into lower and upper triangular matrices is as follows: the matrix is determined to be real or complex, banded or nonbanded, in core or out of core. The decomposition of all matrices is identical, the only differences occurring when a matrix resides out of core. When a matrix is stored on a peripheral file the initial action is to read in as much of the matrix as will fit into the core space allocated. Decomposition proceeds normally until the end of the current incore matrix is reached. At this point the diagonal column is maintained in core and the rest of the available core space is used to read in the next block of columns of the matrix. This proceeds until all of the columns of the matrix have been decomposed. The column elements of the matrix below the diagonal row are written to a separate file as they are decomposed. The pivot row is written to another file as it is decomposed. When the entire matrix has been decomposed for the given diagonal element, the file which has just been written with the decomposed matrix is interchanged with the file upon which the matrix was stored. The decomposition then proceeds identically as before with the files swapped. In this way, only those elements which are needed for further decomposition, that is, those elements to the right of the diagonal element column and beneath the diagonal element row, are written out to the file and consequently read back in for the next round of decomposition. With this method, one eventually gets to the point at which the rest of the matrix to be decomposed resides in core. At this point, the out-of-core decomposition process ceases, and the matrix is decomposed as if it resided entirely in core. The difference between handling a banded versus a nonbanded matrix is in the limits assigned to the rows and columns to be decomposed. The difference between handling a real or complex matrix has to do simply with the number of words per matrix element. Complex matrices have two words per element. At the conclusion of decomposing an out-of-core matrix those parts of the matrix which belong to the lower and upper triangular matrices are written out to the respective files.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
A	In-core storage area available for the matrix
BANDED	Logical variable set to true when decomposing a banded matrix

BNDMAT	Name for a banded matrix
BUBUFR	Storage area to buffer the upper triangular matrix
BUI	The imaginary part of the upper element of the decomposed matrix
BUR	The real component of the upper element of the decomposed matrix
DIAGI	The imaginary part of the diagonal element
DIAGR	The real component of the diagonal element
DMAG	The magnitude of the diagonal element
DMAGSV	Saved value of DMAG for checkpoint/restart
DMAX	Maximum diagonal element
DMIN	Minimum diagonal element
DT	Elapsed time of execution
IBIT	Character value indicating real or complex matrix
IJ	The location of the JK element of the matrix to be decomposed
IJSAV	IJ + NPRELM
IK	Record index for matrix
INCORE	Logical variable set to TRUE when the entire matrix resides in the core
INDX	Statement function to determine the location of the (I, J) element of the matrix to be decomposed
IOA	Logical unit containing the matrix to be decomposed
IOBL	Logical unit upon which the lower decomposed matrix is to be written

DECOMP (MOM)

IOBU	Logical unit on which the upper decomposed matrix is to be written
IOS1	Scratch logical unit designator
IOS1SV	Saved value of IOS1
IOS2	Scratch logical unit designator
IR	Index pointer to upper triangular matrix
IW1	The location of the real part of the diagonal element
IW2	The location of the imaginary part of the diagonal element
JLIM	Upper bound for number of columns in core
JPVT	Index to the pivot column in core
JROW	Location of the pivot row
K1	JPVT + 1
KOL	Pointer to current column in core
LSTCLM	Last column in core to be decomposed
LSTIJ	Location of last element to be decomposed
LSTWRD	Last word to be retrieved for the current operation
M	Matrix bandwidth
MAXBUP	Dimension size of BUBUFR
MAXCOL	Maximum number of columns in core
MAXCOR	Maximum number of words available in core
MOVWRD	Number of words to move a file pointer
MP1	M plus one
MXBAND	Bandwidth plus one at any stage of the decomposition

NAMEA	Matrix name
NBUF	Number of blocks that the submatrix has been written into
NCOL	Number of columns in A
NCORE	Input argument which tells the maximum number of columns that can be put in core
NDIM	Current row dimension of the matrix
NDXL	Index to the NDATBL for the lower triangular matrix
NDXU	Index to the NDATBL for the upper triangular matrix
NMAT	Submatrix number
NPRAIJ	Number of words per element of the matrix to be decomposed
N[illegible]OL	The number of words per column of the matrix to be decomposed
N[illegible]	Internal value for NPRAIJ
N[illegible]LIN	The number of elements to the right of the diagonal element
NPRROW	The number of elements beneath the diagonal
NREAD	Number of columns that have been read from the scratch file
NREADA	Number of columns that have been read from the source file
NRECA	First record to be read into core from A
NROW	Number of rows in matrix A
NUMCOL	Number of columns in matrix A
NWRDS	Number of words to be read from or written to a file
NXTWRD	Next word to be read
PIVRAT	Pivot ratio = DMAX/DMIN

REAL	Logical variable set to TRUE when the matrix to be decomposed is real
REALM	Contains the Hollerith word REAL
SIZE	Temporary location for SQUARE or BANDED
SQUARE	Contains the Hollerith word SQUARE
TLEFT	The amount of time to decompose the matrix
TNOW	Current time
TSTART	Time subroutine TICHEK was last called
TYPE	Type of numbers in the matrix, real or complex

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
A	F.P.
CHKPNT	/SYSFIL/
CHKWRT	/SYSFIL/
CPFRWD	/SYSFIL/
DBGPRT	/ADEBUG/
IOA	F.P.
IOBL	F.P.
IOBU	F.P.
IOCKPT	/SYSFIL/
IOFILE	/IOFLES/
IOS1	F.P.
IOS2	F.P.
LSTSYS	/SYSFIL/
MAXBUP	F.P.
MXBAND	F.P.

NAMEA	F.P.
NCOL	F.P.
NCORE	F.P.
NDXL	F.P.
NDXU	F.P.
NMAT	F.P.
NPRAIJ	F.P.
NROW	F.P.
RSTART	/SYSFIL/
TIMTGO	/SYSFIL/

B. OUTPUT LOCATION

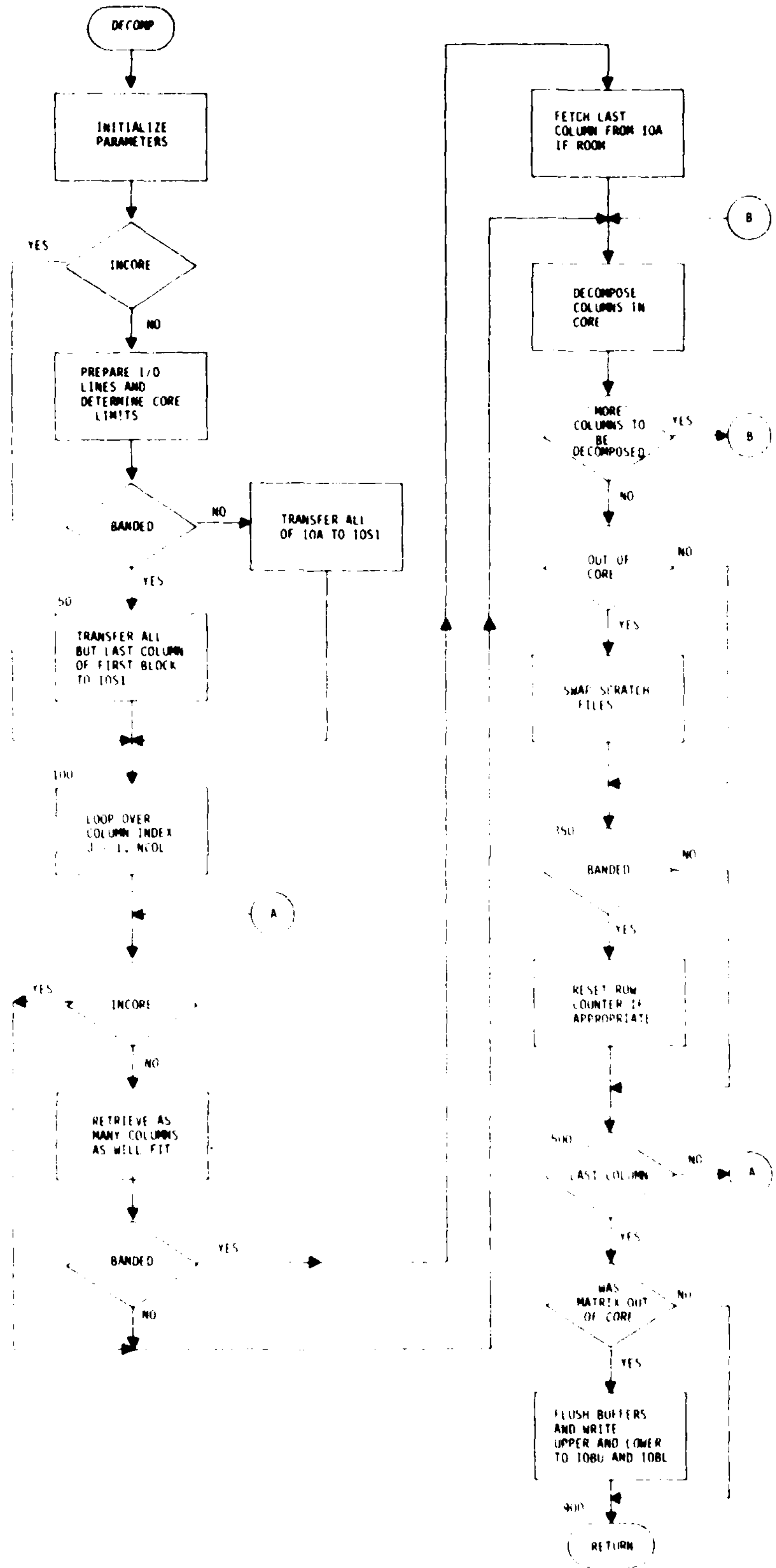
A	F.P.
BUBUFR	F.P.
CHKWRT	/SYSFIL/
LSTSYS	/SYSFIL/
NDATBL	/PARTAB/
RSTART	/SYSFIL/

6. CALLING ROUTINE: LUDDRV

7. CALLED ROUTINES:

ASSIGN	STATIN
CLSFIL	STATOT
ERROR	SYSCHK
GETSYM	TICHEK
MOVFIL	WLKBCK
OPNFIL	WRTFIL
RDEFIL	

DECOMP (MOM)


DECOMP
INITIALIZE PARAMETERS
INCORE
YES
NO
PREPARE I/O LINES AND DETERMINE CORE LIMITS
BANDED
NO
TRANSFER ALL OF IOA TO IOS1
YES
50
TRANSFER ALL BUT LAST COLUMN OF FIRST BLOCK TO IOS1
100
LOOP OVER COLUMN INDEX
A
INCORE
RETRIEVE AS MANY COLUMNS AS WILL FIT
BANDED
FETCH LAST COLUMN FROM IOA IF ROOM
B
DECOMPOSE COLUMNS IN CORE
MORE COLUMNS TO BE DECOMPOSED
OUT OF CORE
SWAP SCRATCH FILES
350
BANDED
RESET ROW COUNTER IF APPROPRIATE
500
LAST COLUMN
WAS MATRIX OUT OF CORE
FLUSH BUFFERS AND WRITE UPPER AND LOWER TO IOBU AND IOBL
900
RETURN

1. NAME: DFPTCL (GTD)

2. PURPOSE: To determine the up-to-four diffraction points which can occur on a cylinder end cap rim for a given radiation direction $\hat{D}$ in the far field or to a specific point FLDPT in the near-field.

3. METHOD: An eighth order polynomial equation is used to solve for eight possible points on the end cap rim that can be diffraction points. These points are defined by elliptic angles in the local elliptic coordinate system for the end cap. The points are next integerized and sorted to remove duplicate points. The accuracy of the possible diffraction points is then improved by a first order Taylor series interpolation scheme. The details are given on pages 125-127 of reference A. The two-to-four correct diffraction points remaining are verified by checking to see which points satisfy the laws of diffraction. An illustration is shown in figure 1.

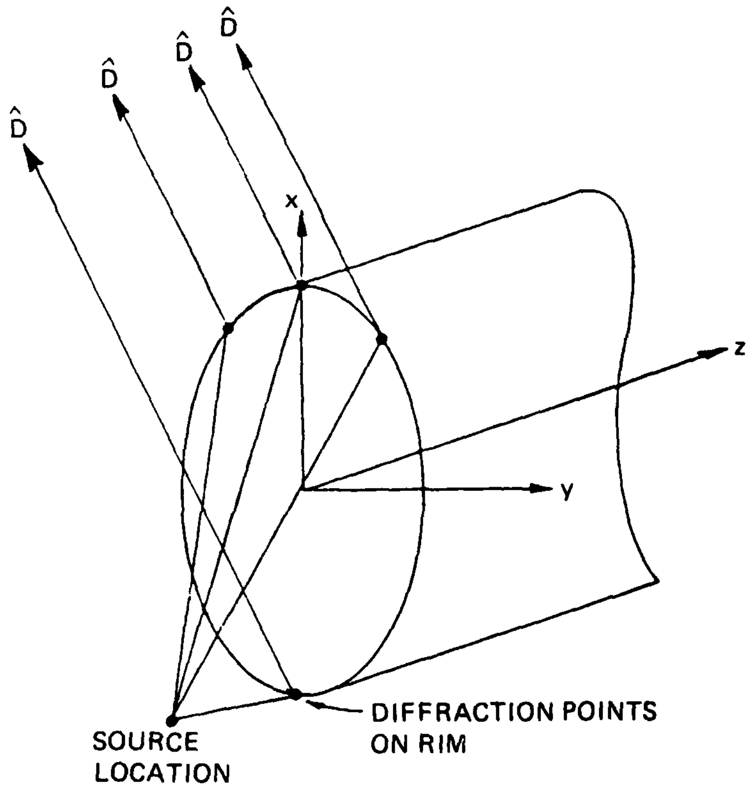


Figure 1. Curved Wedge Diffraction Points on Rim of End Cap of Elliptic Cylinder

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
A	Radius of cylinder along x axis in wavelengths
AE	Half-length of end cap (half-length of line created by intersection of end cap and x-z plane)
B	Radius of cylinder along y axis in wavelengths
C	Cosine of VR
CC	Polynomial equation coefficients
CNC	Cosine of the angle between the z axis and the end cap plane, where CNC(1) refers to the more positive end cap and CNC(2) refers to the more negative end cap (angle measured in x-z plane)
CTC	Cotangent of the angle between the z axis and the end cap plane, where CTC(1) refers to the more positive end cap and CTC(2) refers to the more negative end cap (angle measured in x-z plane)
CV	Computational variable
D	The x, y, z components of the unit vector of the observation propagation direction
D4	Computational variable
DD	Computational variable
DEEX,DEEY,DEEZ	X,Y,Z components of vector from diffraction point to center of end cap in RCS
DEL	Test variable
DEN1	Magnitude of unnormalized edge unit vector
DEN2	Distance from source to improved diffraction point

DEN3	Length of incident ray vector
DEN5	Computational variable
DM	X,Y,Z components of unit vector of propagation direction in end cap coordinate system
DOTQ1	Dot product of edge vector and incident ray
DOTQ2	Dot product of edge vector and diffracted ray
DPR	Conversion factor for converting angular measurements in radians to degrees (180/π)
DSSX,DSSY,DSSZ	X,Y,Z components of vector tangent to diffraction point in end cap plane in RCS
DV	Change in elliptic angle V calculated to improve accuracy of diffraction point
EEX,EEY,EEZ	X,Y,Z components of ray tangent to diffraction point in RCS
EPSQ	Difference in DOTQ1 and DOTQ2 (error test variable)
EXQ,EYQ,EZQ	X,Y,Z components of normalized edge unit vector in RCS
FLDPT	The x,y,z components of the near-field field point location in the reference coordinate system in wavelengths
I	Do loop variable
IDEL	Test variable
IV	Elliptic angles defining permissible diffraction points in RCS x-y plane (in degrees, rounded off to nearest integer)
J	Elliptic angle defining diffraction point in RCS x-y plane in degrees
K	Do loop variable

LNRFLD	Flag which indicates if far-field (LNRFLD=0) or near-field (LNRFLD=1) calculations were requested
N	Index variable (also number of permissable roots)
NC	End cap where diffraction occurs
NCC	Sign change variable
P	Polynomial equation variable
Q	Polynomial equation variable
QC	Complex conjugate of Q
R	Polynomial equation variable
RC	Complex conjugate of R
ROOT	Roots of polynomial equation returned from subroutine POLYRT
RPD	Conversion factor for converting angular measurements in degrees to radians ($\pi/180$)
S	Sine of elliptic angle V (also polynomial equation variable)
SNC	Sine of the angle between the z axis and end cap plane, where SNC(1) refers to the more positive end cap and SNC(2) refers to the more negative end cap (angle measured in x-z plane)
SSX,SSY,SSZ	X,Y,Z components of vector incident on edge in RCS
SXQ,SYQ,SZQ	X,Y,Z components of unit vector of propagation direction of incident ray in RCS
V	Elliptic angles defining diffraction points in RCS x-y plane
VQ	Elliptic angle defining diffraction point (improved accuracy)
VR	Elliptic angle defining diffraction point

VT	Elliptic angle defining diffraction point (improved accuracy) in degrees
XS	The x,y,z components of the source location in the reference coordinate system in wavelengths
XSM,YSM,ZSM	X,Y,Z components of source location in end cap coordinate system
ZC	Point where end cap intersects z axis of reference coordinate system, where ZC(1) refers to the more positive end cap end ZC(2) refers to the more negative end cap

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
A	/GEOMEL/
B	/GEOMEL/
CNC	/GEOMEL/
CTC	/GEOMEL/
D	/DIR/
DPR	/PIS/
FLDPT	/NEAR/
LNRFLD	/NEAR/
NC	F.P.
RPD	/PIS/
SNC	/GEOMEL/
ZC	/GEOMEL/

B. OUTPUT

OUTPUT	LOCATION
V	F.P.

6. CALLING ROUTINE:

   ENDIF

7. CALLED ROUTINES:

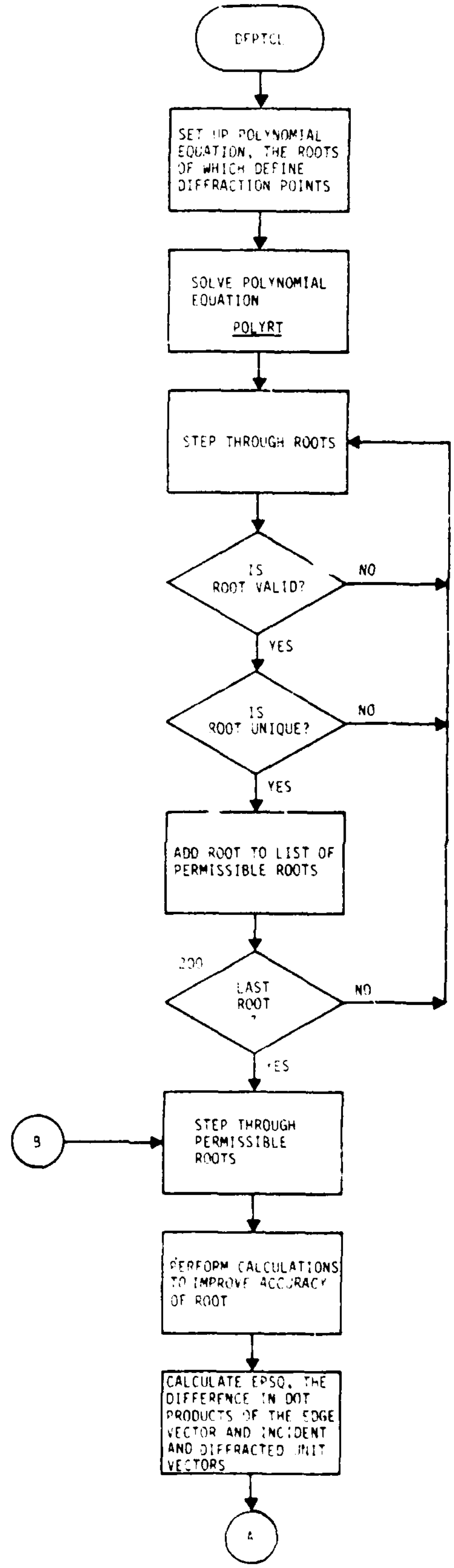
   BABS

   NFD

   POLYRT

8. REFERENCES:

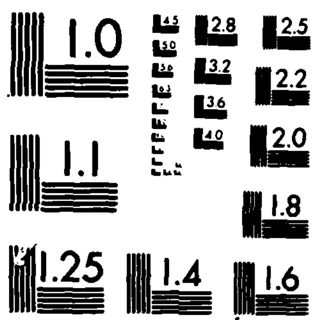
A. R. J. Marhefka, "Analysis of Aircraft Wing-Mounted Antenna Patterns," Report 2902-25, June 1976, The Ohio State University ElectroScience Laboratory, Department of Electrical Engineering; prepared under Grant No. NGL 36-008-138 for National Aeronautics and Space Administration.


DFPTCL
SET UP POLYNOMIAL EQUATION, THE ROOTS OF WHICH DEFINE DIFFRACTION POINTS
SOLVE POLYNOMIAL EQUATION
POLYRT
STEP THROUGH ROOTS
IS ROOT VALID?
NO
YES
IS ROOT UNIQUE?
NO
YES
ADD ROOT TO LIST OF PERMISSIBLE ROOTS
200
LAST ROOT ?
NO
YES
B
STEP THROUGH PERMISSIBLE ROOTS
PERFORM CALCULATIONS TO IMPROVE ACCURACY OF ROOT
CALCULATE EPSQ, THE DIFFERENCE IN DOT PRODUCTS OF THE EDGE VECTOR AND INCIDENT AND DIFFRACTED UNIT VECTORS
A

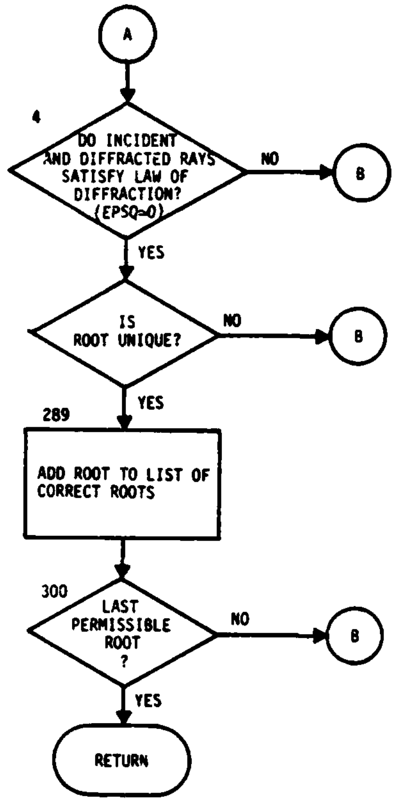
AD-A137 463 GENERAL ELECTROMAGNETIC MODEL FOR THE ANALYSIS OF COMPLEX SYSTEMS (GEMACS..(U) BDM CORP ALBUQUERQUE NM D L KADLEC ET AL. SEP 83 BDM/A-83-020-TR

3/5

UNCLASSIFIED RADC-TR-83-217-VOL-3 F30602-81-C-0084 F/G 20/14 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A


A
4
DO INCIDENT AND DIFFRACTED RAYS SATISFY LAW OF DIFFRACTION? (EPSQ≈0)
NO
B
YES
IS ROOT UNIQUE?
NO
B
YES
289
ADD ROOT TO LIST OF CORRECT ROOTS
300
LAST PERMISSIBLE ROOT ?
NO
B
YES
RETURN

1. NAME: DFPTWD (GTD)

2. PURPOSE: To determine the diffraction point on edge ME of plate MP for a given source location $\overline{XS}$ and a diffracted ray direction $\hat{D}$ for far field or a field point FLDX for near-field calculations.

3. METHOD: For far-field calculations the cosine of β_o, the diffraction angle, is known by:

$$\cos \beta_o = \hat{D} \cdot \hat{V}$$

where $\hat{D}$ is the given far-field direction and $\hat{V}$ is the plate edge unit vector. Then the cosines of the angle between the plate edge and the vector from the plate corners to the source are found. The cosine of the diffraction angle must fall between these two angle cosines or the diffraction point is not on the plate edge. If it does not fall within the corners, LDIFFR is set to false to indicate to the calling routine that diffraction did not occur. If the diffraction angle is between the bounds, the diffraction point location is found. Figure 1 shows the important far-field geometry.

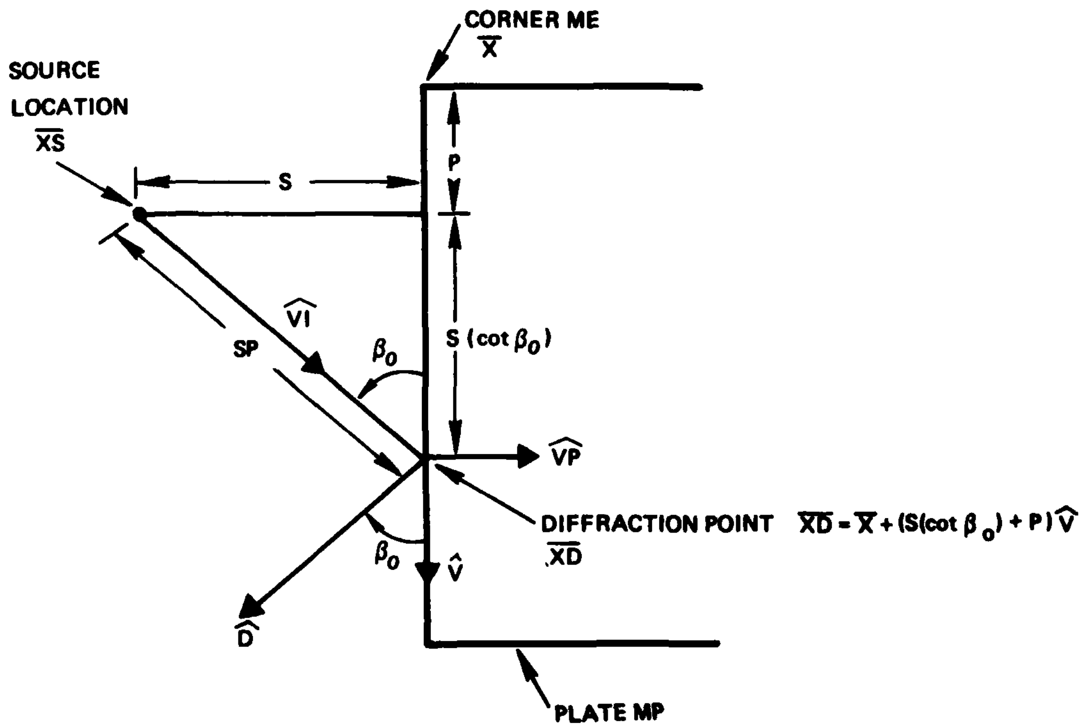


Figure 1. Geometry Used in Defining Far-Field Diffraction Point on Plate Edge

The steps needed to find a near-field diffraction point are more involved. Figures 2 and 3 show the needed near-field geometry.

$\overline{S_e'}$ is the vector from the edge-fixed coordinate system origin* to the source point. To calculate $\overline{S_e'}$ in the edge-fixed coordinate system, first find $\overline{S_e'}$ in the reference coordinate system (RCS). $\overline{S_{e,RCS}'}$ is given by

$$\overline{S'_{e,RCS}} = \overline{XS} - \overline{X}.$$

Then dot $\overline{S_{e,RCS}'}$ with the unit vectors that represent the edge-fixed coordinate system. These unit vectors are VP, VN and V. VP, the plate binormal, is the RCS representation of X_e. VN, the plate normal, is the RCS representation of Y_e. V, the plate edge unit vector, is the RCS representation of Z_e. This calculation is written as

$$S_e'(1) = \overline{S'_{e,RCS}} \cdot VP$$

$$S_e'(2) = \overline{S'_{e,RCS}} \cdot VN$$

$$S_e'(3) = \overline{S'_{e,RCS}} \cdot \hat{V}$$

$\overline{S_e'}$ in the edge-fixed coordinate system is given as

$$\overline{S_e'} = S_e'(1)\,\hat{X}_e + S_e'(2)\,\hat{Y}_e + S_e'(3)\,\hat{Z}_e.$$

*The edge-fixed coordinate system is a coordinate system applied to a plate edge such that the x_e value is in the plate plane and normal to the edge (the edge binormal), the y_e value is normal to the plate and the z_e value is along the plate edge. For diffraction calculations, the origin of this system is set at the edge corner from which a ray in the direction of z_e would propagate away.

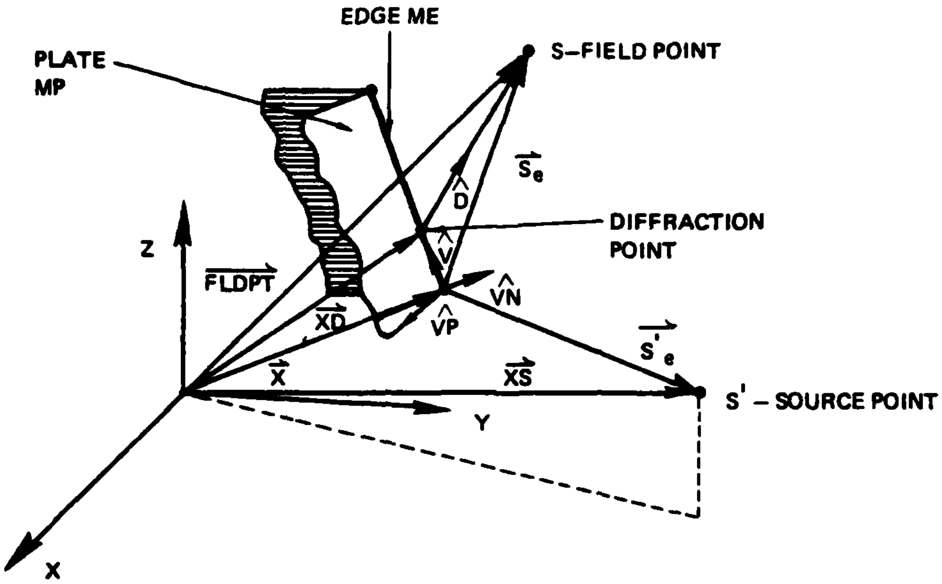


Figure 2. RCS and Edge-Fixed Coordinate System Geometry for Calculation of Near-Field Diffraction Point

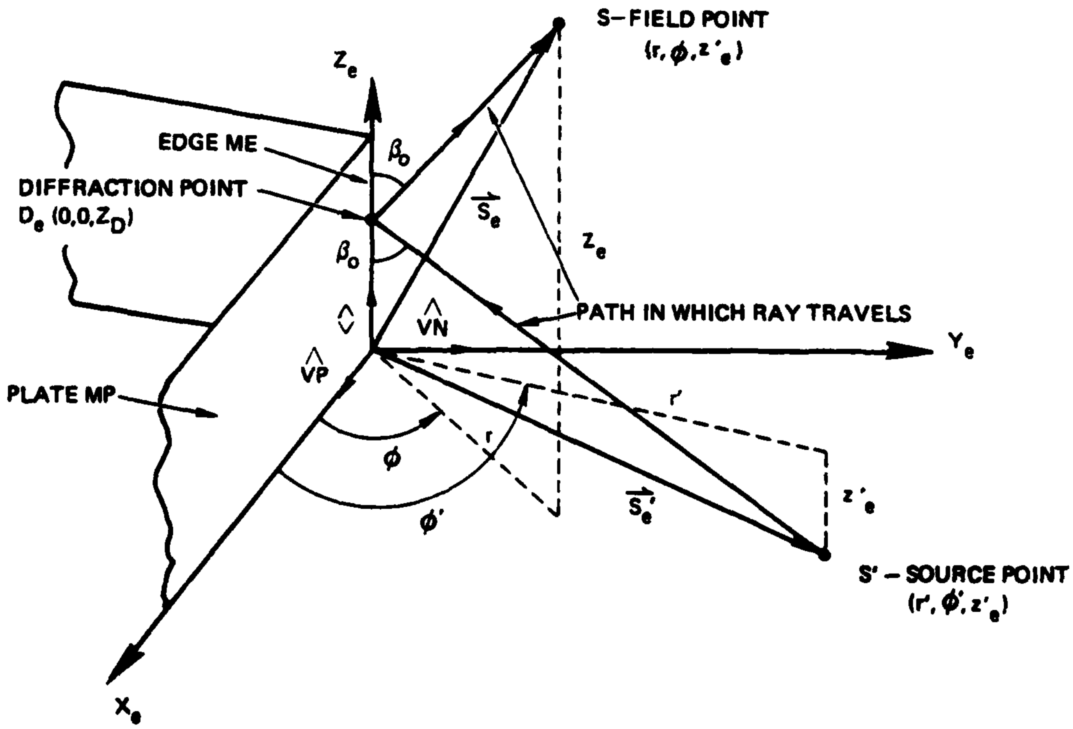


Figure 3. Edge-Fixed Coordinate System Geometry for Calculation of Near-Field Diffraction Point

$\overline{S_e}$ is the vector from the edge-fixed coordinate system origin to the field point. To calculate $\overline{S_e}$ in the edge-fixed coordinate system, first find $\overline{S_e}$ in the RCS. $\overline{S_{e,RCS}}$ is given by

$$\overline{S_{e,RCS}} = \overline{FLDPT} - \overline{X}.$$

Then dot $\overline{S_{e,RCS}}$ with the edge-fixed coordinate system unit vectors from the RCS. This gives

$$S_e(1) = \overline{S_{e,RCS}} \cdot \widehat{VP}$$

$$S_e(2) = \overline{S_{e,RCS}} \cdot \widehat{VN}$$

$$S_e(3) = \overline{S_{e,RCS}} \cdot \hat{V}$$

$\overline{S_e}$ in the edge-fixed coordinate system is given by

$$\overline{S_e} = S_e(1)\ \hat{X}_e + S_e(2)\ \hat{Y}_e + S_e(3)\ \hat{Z}_e$$

Now the values of r', z_e', r and z_e are needed. The value z_e' is $S_e'(3)$, the edge-fixed coordinate system z value of the source point. The value z_e is $S_e(3)$, the edge-fixed coordinate system z value of the field point. The value r' is found by

$$r' = \sqrt{x_e'^2 + y_e'^2}$$

where x_e' is $S_e'(1)$ and y_e' is $S_e'(2)$, the x and y edge-fixed coordinate system values of the source point. The value r is found by

$$r = \sqrt{x_e^2 + y_e^2}$$

where x_e is $S_e(1)$ and y_e is $S_e(2)$, the edge-fixed coordinate system x and y values of the field point.

The diffraction angle, β_o, is found as:

$$\beta_o = \text{arc tan} \left(\frac{r' + r}{z_e - z_e'} \right)$$

The diffraction point location on the edge, Z_D, is given as

$$z_D = z_e - \frac{r}{\tan \beta_o} .$$

The edge-fixed coordinate system vector from the origin to the diffraction point, $\overline{D_e}$, is defined as

$$\overline{D_e} = 0 \hat{X}_e + 0 \hat{Y}_e + z_D \hat{Z}_e .$$

The diffraction point in RCS coordinates is determined by the vector from the RCS origin to the diffraction point, $\overline{XD}$. This vector is found by the addition of the vector from the RCS origin to the edge-fixed coordinate system origin, $\overline{X}$, and $\overline{D_e}$ in RCS coordinates. $\overline{D_{e,RCS}}$ is obtained by multiplying Z_D and V. $\overline{XD}$ is given as

$$\overline{XD} = \overline{X} + z_D \hat{V} .$$

Two checks are required to verify that diffraction did occur. For diffraction to occur, the diffraction point must be on the finite plate edge. If the edge length is VMAG, then $0 \leq Z_D \leq$ VMAG for the diffraction point to be on the edge. Also the dot product of $\hat{D}$, the observation direction, and $\hat{V}$, the plate edge vector needs to be checked. $\hat{D} \cdot \hat{V}$ is the cosine of β_o, the diffraction angle. If the absolute value of $\hat{D} \cdot \hat{V}$ is less than 0.999, then diffraction did occur. If it is greater than 0.999, diffraction is said not to have occurred because it is too close to grazing incidence.

After the diffraction point location for far field or near field has been found and verified, the distance from the source to the diffraction point (SP) and the incident ray unit vector ($\hat{VI}$) are determined by

$$SP = \sqrt{\overline{XD} - \overline{XS}}$$

$$\hat{VI} = \frac{\overline{XD} - \overline{XS}}{SP} .$$

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
B	The diffraction angle β_o
BDHI	Upper bound cosine for a permissible diffraction angle
BDLOW	Lower bound cosine for a permissible diffraction angle
BRD	Array for the dot products of the unit edge vector and the unit vector between the source and each corner on edge ME
CTB	Cotangent of B
D	Observation direction
DV	Cosine of B
EMAG	The length of plate edge ME
FLDX	Near-field observation point
LCORNR	Logical variable set true if corner diffraction calculations were requested
LDIFFR	Logical variable. Set true to indicate if diffraction occurs. Set false to indicate diffraction not possible

LNRFLD	Flag which indicates if far-field (LNRFLD=0) or near-field (LNRFLD=1) calculations were requested
ME	Edge where diffraction occurs; also first corner on this edge
MEP	Array which contains the number of edges (or corners) on plate MP
MP	Plate where diffraction occurs
N	DO loop variable
P	Dot product of edge vector and vector from corner ME to source
RE	The x_e-y_e plane magnitude of the field point vector in the edge-fixed coordinate system
RPE	The x_e-y_e plane magnitude of the source point location vector in the edge-fixed coordinate system
S	Perp.ndicular distance from source to edge ME
SE	The vector from the edge-fixed coordinate system origin to the field point in the edge-fixed coordinate system
SERCS	The vector from the edge-fixed coordinate system origin to the field point in RCS coordinates
SP	Distance from source to diffraction point
SPE	The vector from the edge-fixed coordinate system origin to the source point in edge-fixed coordinate system values
SPERCS	The vector from the edge-fixed coordinate system origin to the source point in RCS coordinates
SX	Variable used to calculate S
V	Array which contains the edge unit vector for edge ME of plate MP in RCS

VC	The x, y, z components of the two unit vectors from the source to the corners on edge ME
VCM	Array which contains the distance between the source and each corner on edge ME
VI	Incident ray unit vector
VN	Array which contains the unit normal for plate MP in RCS
VP	Array which contains the unit binormal for edge ME of plate MP in RCS
X	Array which contains the corner locations for plate MP edge ME in the reference coordinate system
XD	Location of diffraction point
XS	Source location
ZD	The z_e direction value of the diffraction point on the plate edge in the edge-fixed coordinate system
ZE	The z_e direction value of the field point in the edge-fixed coordinate system
ZPE	The z_e direction value of the source location in the edge-fixed coordinate system

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
D	F.P. (for far field)
FLDX	F.P.
LCORNR	/LOGDIF/
LNRFLD	/NEAR/
ME	F.P.
MEP	/GEOPLA/

	MP	F.P.
	V	/GEOPLA/
	VN	/GEOPLA/
	VP	/GEOPLA/
	X	/GEOPLA/
	XS	F.P.
B.	OUTPUT	LOCATION
	BRD	F.P.
	D	F.P. (for near field)
	DV	F.P.
	LDIFFR	F.P.
	SP	F.P.
	VCM	F.P.
	VI	F.P.
	XD	F.P.

6. CALLING ROUTINES:

DIFPLT

DPLRPL

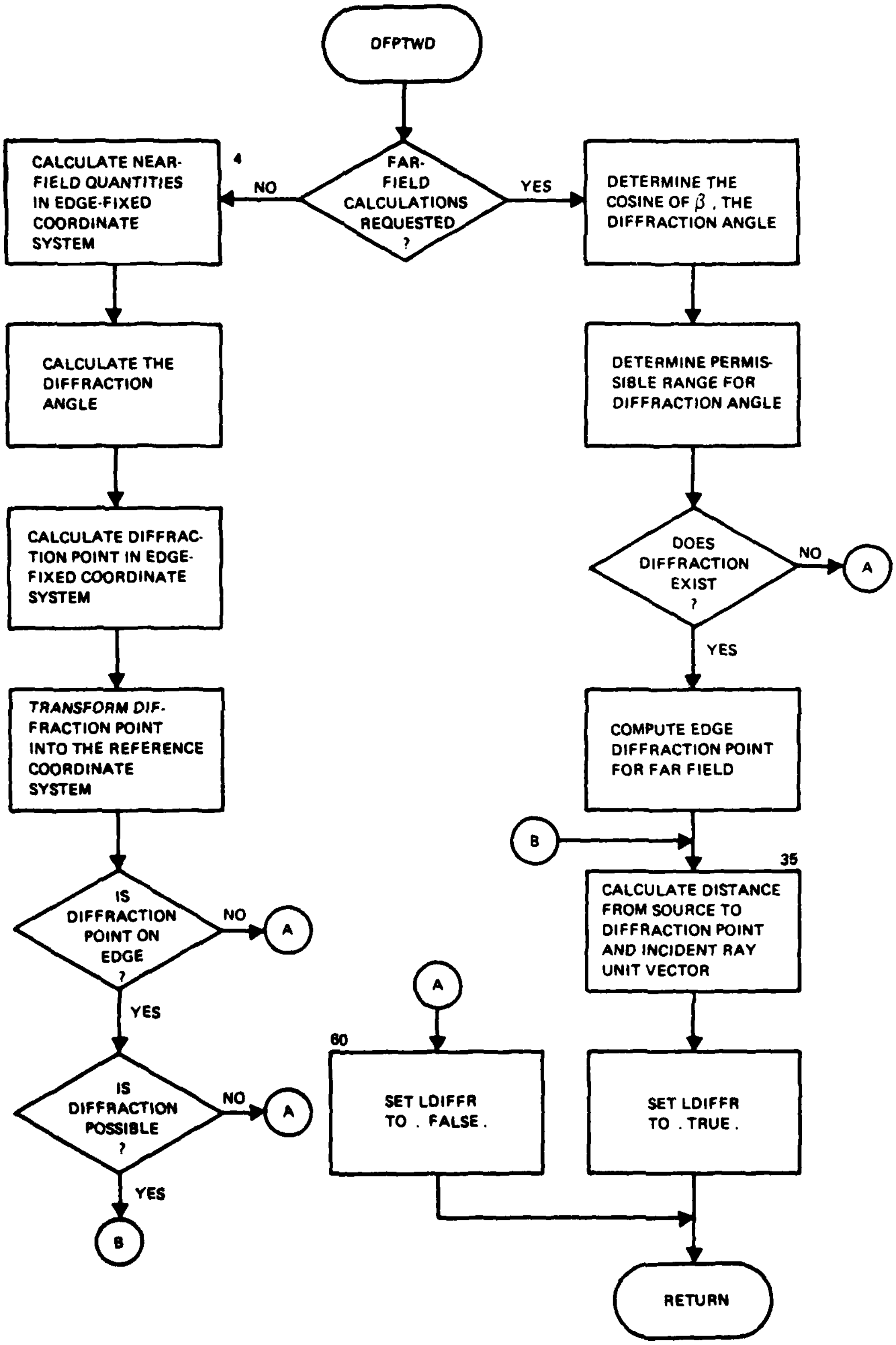
RFDFPT

RPLDPL

7. CALLED ROUTINES:

BTAN2

NFD


DFPTWD
FAR-FIELD CALCULATIONS REQUESTED ?
NO
YES
4
CALCULATE NEAR-FIELD QUANTITIES IN EDGE-FIXED COORDINATE SYSTEM
DETERMINE THE COSINE OF β, THE DIFFRACTION ANGLE
CALCULATE THE DIFFRACTION ANGLE
DETERMINE PERMISSIBLE RANGE FOR DIFFRACTION ANGLE
CALCULATE DIFFRACTION POINT IN EDGE-FIXED COORDINATE SYSTEM
DOES DIFFRACTION EXIST ?
NO
A
YES
TRANSFORM DIFFRACTION POINT INTO THE REFERENCE COORDINATE SYSTEM
COMPUTE EDGE DIFFRACTION POINT FOR FAR FIELD
B
35
IS DIFFRACTION POINT ON EDGE ?
NO
A
CALCULATE DISTANCE FROM SOURCE TO DIFFRACTION POINT AND INCIDENT RAY UNIT VECTOR
A
YES
60
IS DIFFRACTION POSSIBLE ?
NO
A
SET LDIFFR TO .FALSE.
SET LDIFFR TO .TRUE.
YES
B
RETURN

1. NAME: DFRFPT (GTD)

2. PURPOSE: To determine the ray path for a source ray which is diffracted off a given edge on a given plate and then reflected in a given direction by a cylinder.

3. METHOD: The diffraction point on a plate edge and the reflection point on an elliptic cylinder for a diffracted-reflected ray in a given observation direction are calculated via an iterative process. Pertinent geometry is shown in figure 1. To avoid the 2π-to-0 transition in ϕ, the reference ϕ value is rotated to place this branch cut behind the cylinder (shadowed from the plate edge), and the angular variable ϕ' is used in this iteration process (see figure 2).

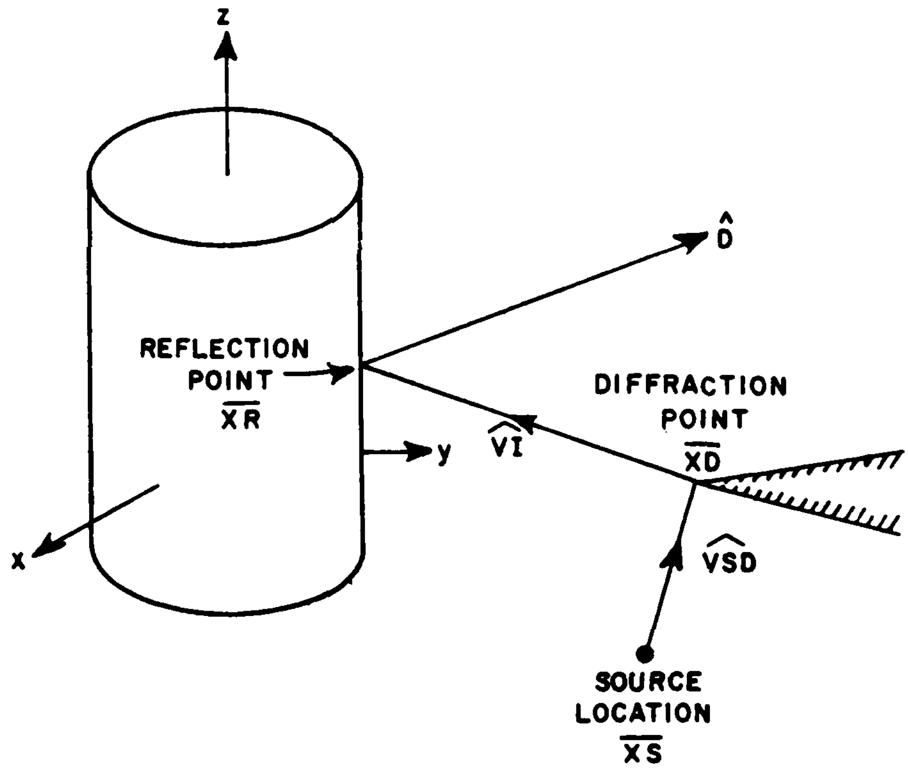


Figure 1. Geometry Used in DFRFPT for Ray Diffracted by Plate and Then Reflected by the Cylinder

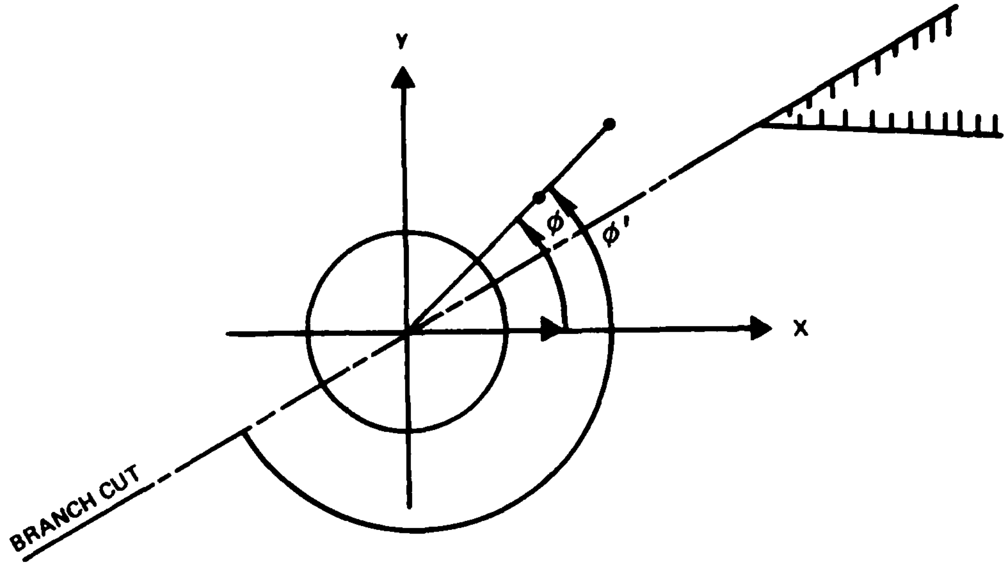


Figure 2. Illustrating Placement of Branch Cut to Avoid 2π-to-0 Transition in ϕ

The iteration begins with an initial diffracted-reflected ray which satisfies the laws of diffraction and reflection. Starting data are obtained in one of two ways. If a previous call to this routine (for the same plate edge and source) has successfully found the diffracted-reflected ray path, this previous path is used as starting data. Otherwise the starting reflection point is defined on the rim of the cylinder closest to the plate edge under consideration. Then the corresponding diffraction point is found by enforcing Snell's law along the plate edge. The first method is preferable to the second since, in general, far-field ray directions ($\hat{D}$) in subsequent calls to DFRFPT will not differ greatly. For example, in calculating a far-field pattern cut, the far-field θ and ϕ angles will differ by only a few degrees, and the closeness of the starting point will lead to fewer iterations in order to obtain convergence.

The path of the starting ray defines the initial cylinder reflection point ($\overline{XR}$) and the edge diffraction point ($\overline{XD}$). In almost every instance, the resulting far-field radiation direction of this ray will <u>not</u> be the desired radiation direction. The angular difference between (θ, ϕ) of the starting ray (THOR, PHORP) and the (θ, ϕ) of the desired far-field direction (THSR, PHSRP) is divided into a number of small angular steps ($\Delta\theta$, $\Delta\phi$) = (DTSR, DPSR). This iteration moves the reflection and diffraction points from their initial positions in small steps corresponding to angular changes ($\Delta\theta$, $\Delta\phi$) so that when the iteration is complete the resulting $\overline{XR}$ and $\overline{XD}$ will define the reflection and diffraction points that give a far-field D-directed ray. The number of steps to be taken (IVD) is determined from the starting data. Should convergence not be reached in IVD steps, the number of steps is doubled (up to 32 steps) and the iteration repeated. The doubling process is the outer loop of the flow chart. Should convergence be reached with IVD steps and the Snell's law error be significantly smaller than required, IVD for the plate and edge under consideration is halved prior to exiting the routine.

The iterations which step through the (θ, ϕ) angles by $(\Delta\theta, \Delta\phi)$ correspond to the inner (DO 50) loop of the flow chart. Each iteration has three steps:

(a) Compute the diffraction point ($\overline{XD}$) from known reflection point ($\overline{XR}$), source point ($\overline{XS}$) and edge unit vector ($\overline{V}$). This is done by a simple application of Snell's law.

(b) The change in cylinder elliptic angle (DV) and z coordinate (DU) are computed from a Taylor series expansion. The expansion requires the calculation of functions and partial derivatives of equations defining elliptic angle (VR) and z coordinate (UR) in terms of far-field angles (θ,ϕ). The equations are given in reference A.

(c) The coordinates of $\overline{XR}$ are computed from the new values of UR and VR.

At the end of the prescribed number of iterations the initial far-field direction has been stepped slowly to the desired far-field direction, and the initial reflection-diffraction points have been stepped from their initial values to candidate reflection-diffraction points. Snell's law is then applied to the final reflection and final diffraction points to see if they qualify as the bona fide ray path. If the error is sufficiently small, the outer loop is exited. Otherwise, the number of steps is doubled, as described above. Should the routine not converge within 32 steps (the maximum number), a warning message is printed on LUPRNT.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
A	Radius of the elliptical cylinder along the x axis of the cylinder, in wavelengths
B	Radius of the elliptical cylinder along the y axis of the cylinder, in wavelengths

CSCE	Dot product of ray from corner of edge ME to source and edge unit vector
D	Propagation direction unit vector in RCS
DDC	This array contains the cosine of the starting reflected ray theta angle, where DDC (MP, ME, N) = COS (TDCR (MP, ME, N))
DE	Dot product of incident ray propagation direction and unit edge vector of edge ME
DOTP	Test variable used to insure reflection was computed properly
DPSR	Phi angle increment size
DR	Reflected ray propagation direction
DRP	X,Y components of phi polarization unit vector for field reflected from cylinder in RCS
DRT	X,Y,Z components of theta polarization unit vector for field reflected from cylinder
DTSR	Theta angle increment size
DU	Change in UR for one iteration using Taylor series expansion
DV	Change in VR for one iteration using Taylor series expansion
ERC	Error detection variable
ERCA	Error detection variable for reflection point
ERCB	Error detection variable for diffraction point
FI	Equation governing the law of reflection
FP	Partial derivative of FI with respect to phi
FU	Partial derivative of FI with respect to UR

FV	Partial derivative of FI with respect to VR
GI	Equation governing the law of reflection
GP	Partial derivative of GI with respect to phi
GT	Partial derivative of GI with respect to theta
GU	Partial derivative of GI with respect to UR
GV	Partial derivative of GI with respect to VR
IVD	Number of steps used in iteration
LDRC	Set true if starting point data are available from previous pattern angle
ME	Edge on plate MP where diffraction occurs
MJ	Starting point corner number
MP	Number of plate where diffraction occurs
PDCR	This array contains angles PDCR (MP, ME, N) defining the phi component of the reflected ray direction of rays diffracted by edge ME of plate MP and then reflected at the starting point N on the cylinder
PHCR	Phi component of reflected ray direction
PHOR	Phi component of reflected ray direction from previous time DFRFPT was called (or present value for next time routine is called)
PHORP	Phi angle of reflected ray direction in rotated RCS system (branch cut placed behind cylinder)
PHSR	Phi angle of the propagation direction
PHSPR	Phi angle of reflected ray direction in rotated RCS system (branch cut placed behind cylinder)

PHWR	The phi angle location of the center of edge ME of plate MP with respect to the cylinder. It is used to specify the branch cut displacement angle for the plate-diffracted, cylinder-reflected terms
PI	π
SNM	Magnitude of unnormalized cylinder normal
SNPX	Partial derivative of SNX with respect to VR
SNPY	Partial derivative of SNY with respect to VR
SNX, SNY	X and Y components of normal to cylinder in RCS components
STP	Number of steps used in iteration
TDCR	This array contains angles TDCR (MP, ME, N) defining the reflected ray theta angle of ray directions for rays diffracted by edge ME of plate MP and then reflected by starting reflection point N on the cylinder
THCR	Theta component of reflected ray direction
THOR	Theta component of reflected ray direction from previous time DFRFPT was called (or for next time routine is called)
THSR	Theta angle of the propagation direction
TPI	2π
UDC	This array contains the linear value UDC(N) defining the z component of the starting reflection points on the cylinder axis. UDC(1) is for the more positive z location and UDC(2) is for the more negative z location
UR	Z coordinate of the cylinder reflection point
URO	Z component of starting reflection point location on cylinder

V	Edge unit vectors for all edges of all plates
VDC	This array contains the elliptical angle VDC(MP,ME) defining the starting reflection point on the cylinder for a ray diffracted from edge ME of plate MP and then reflected by the cylinder
VI	Unit vector of incident ray on cylinder
VIM	Distance from diffraction point to reflection point
VIU	Partial derivative of VI with respect to UR
VIV	Partial derivative of VI with respect to VR
VR	Elliptical angle defining reflection point on cylinder (2-D)
VRO	Elliptical angle defining starting reflection point on cylinder
VSD	X,Y,Z components of propagation vector of ray from source to diffraction point
VSDM	Distance from source to the diffraction point
X	Corner x,y,z coordinates of all plates in RCS
XD	X,Y,Z components of diffraction point location
XP	Point along line drawn through edge ME closest to source.
XR	X,Y,Z components of reflection point location on cylinder
XRU	Partial derivative of XR with respect to UR
XRV	Partial derivative of XR with respect to VR in RCS
XS	X,Y,Z components of the source location in wavelengths in the RCS

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
A	/GEOMEL/
B	/GEOMEL/
D	/DIR/
DDC	/BNDDCL/
LDRC	F.P.
ME	F.P.
MP	F.P.
PDCR	/BNDDCL/
PHSR	/DIR/
PHWR	/BRNPHW/
PI	/PIS/
TDCR	/BNDDCL/
THSR	/DIR/
TPI	/PIS/
UDC	/BNDDCL/
V	/GEOPLA/
VDC	/BNDDCL/
X	/GEOPLA/
XS	/SORINF/

B. OUTPUT

OUTPUT	LOCATION
DE	F.P.
DOTP	F.P.

LDRC	F.P.
SNM	F.P.
VI	F.P.
VIM	F.P.
VR	F.P.
VSD	F.P.
VSDM	F.P.
XD	F.P.
XR	F.P.

6. CALLING ROUTINE:

   DPLRCL

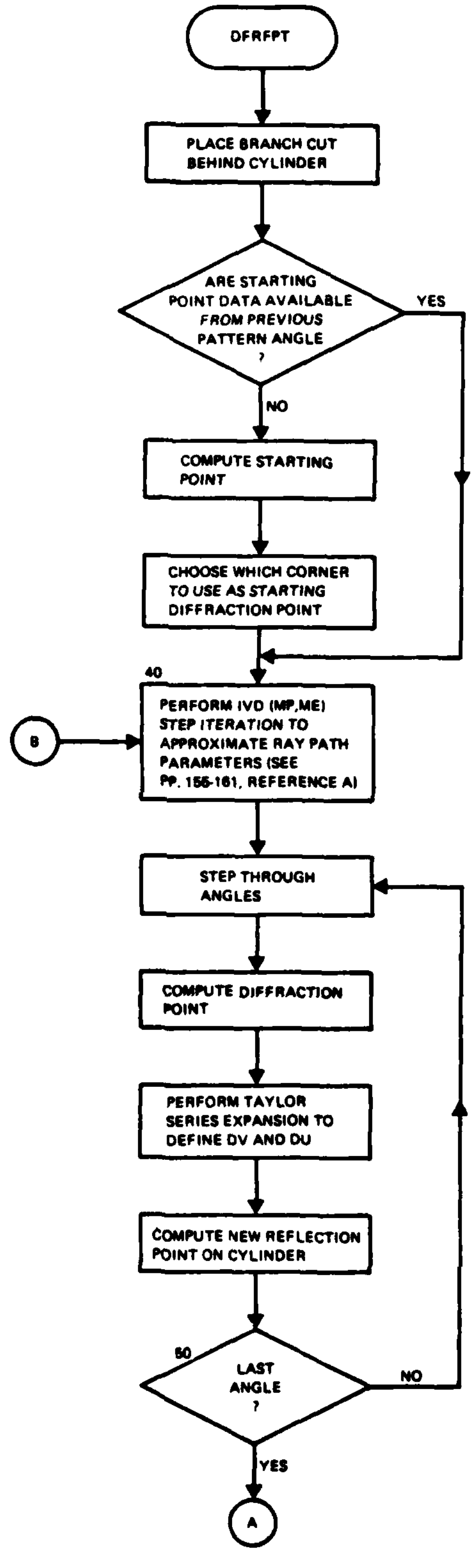
7. CALLED ROUTINE:

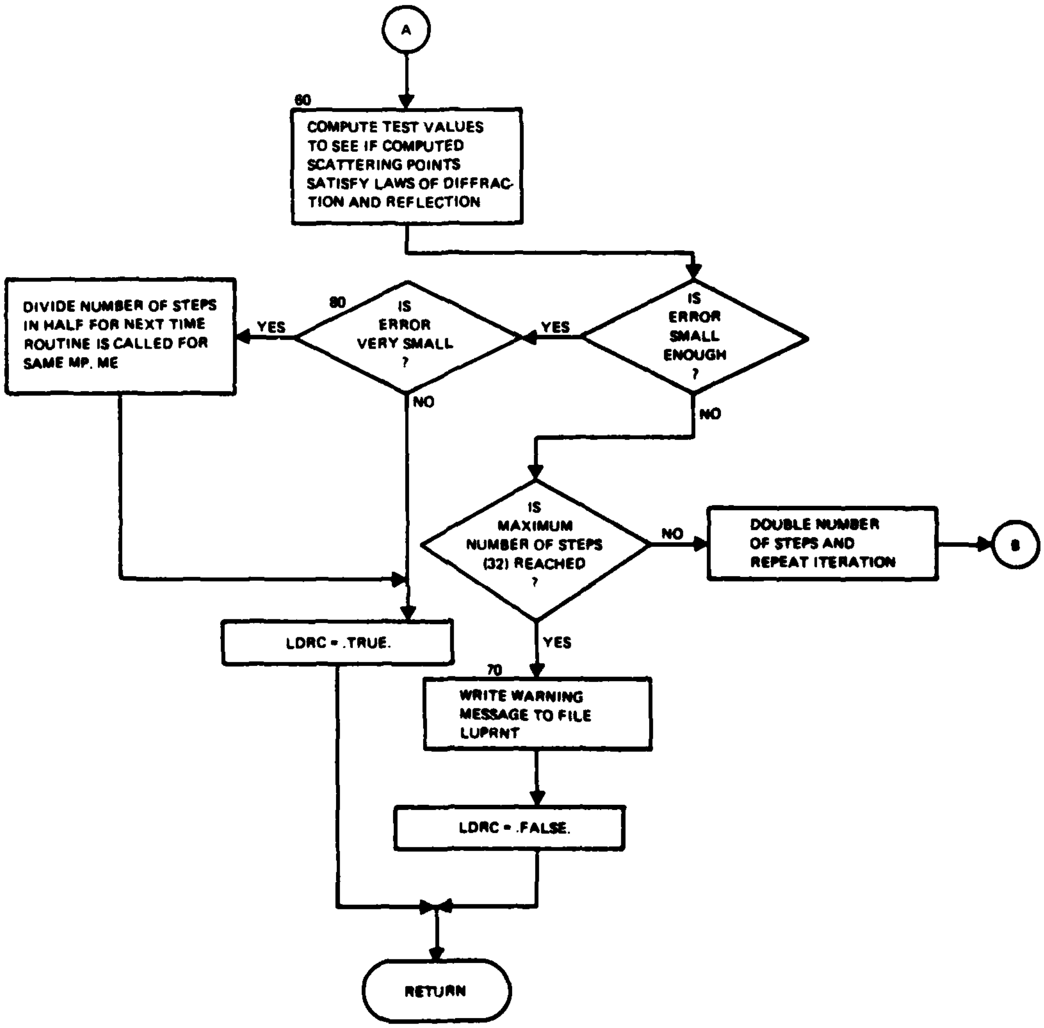
   None

8. REFERENCE:

A. R.J. Marhefka, "Analysis of Aircraft Wing-Mountd Antenna Patterns," Report 2902-25, June 1976, The Ohio State University ElectroScience Laboratory, Department of Electrical Engineering; prepared under Grant No. NGL 36-008-138 for National Aeronautics and Space Administration.

DFRFPT (GTD)


DFRFPT
PLACE BRANCH CUT BEHIND CYLINDER
ARE STARTING POINT DATA AVAILABLE *FROM* PREVIOUS PATTERN ANGLE ?
YES
NO
COMPUTE STARTING POINT
CHOOSE WHICH CORNER TO USE AS STARTING DIFFRACTION POINT
40
PERFORM IVD (MP,ME) STEP ITERATION TO APPROXIMATE RAY PATH PARAMETERS (SEE PP. 156-161, REFERENCE A)
B
STEP THROUGH ANGLES
COMPUTE DIFFRACTION POINT
PERFORM TAYLOR SERIES EXPANSION TO DEFINE DV AND DU
COMPUTE NEW REFLECTION POINT ON CYLINDER
50
LAST ANGLE ?
NO
YES
A


A
60
COMPUTE TEST VALUES TO SEE IF COMPUTED SCATTERING POINTS SATISFY LAWS OF DIFFRACTION AND REFLECTION
IS ERROR SMALL ENOUGH ?
YES
80
IS ERROR VERY SMALL ?
YES
DIVIDE NUMBER OF STEPS IN HALF FOR NEXT TIME ROUTINE IS CALLED FOR SAME MP, ME
NO
NO
IS MAXIMUM NUMBER OF STEPS (32) REACHED ?
NO
DOUBLE NUMBER OF STEPS AND REPEAT ITERATION
B
YES
LDRC = .TRUE.
70
WRITE WARNING MESSAGE TO FILE LUPRNT
LDRC = .FALSE.
RETURN

1. NAME: DICOEF (GTD)

2. PURPOSE: To calculate the incident part or the reflection part of the wedge diffraction coefficient or the corner diffraction coefficient.

3. METHOD: This subroutine computes either the incident part or the reflection part (depending on the value of the input variable BET) of the wedge or corner diffraction coefficient (depending on the value of the logical input variable LOG). The Uniform Geometrical Theory of Diffraction (see reference A) has been used to derive these terms. For wedge diffraction (LOG = FALSE). The coefficient is given as:

$$\underbrace{DICOEF(R,\beta,\sin\beta_o,n)}_{DIR} = \underbrace{\frac{-e^{-j\pi/4}}{2n\sqrt{2\pi k}\,\sin\beta_o}}_{\underbrace{\frac{TOP}{DEM} = COM}} \left\{ \underbrace{\cot\left(\frac{\pi+\beta}{2n}\right) F\left[kRa^{+}(\beta)\right]}_{\underbrace{COTA*FA}} + \underbrace{\cot\left(\frac{\pi-\beta}{2n}\right) F\left[kRa^{-}(\beta)\right]}_{\underbrace{COTA*FA}} \right\},$$

TOP/DEM = COM: $k=2\pi$, since units are normalized to wavelenghts; thus $\sqrt{2\pi k}\;\; 2\pi$

First COTA*FA: Used in UPPI calculation for N^{+} part of diffraction coefficient

Second COTA*FA: Used in UNPI calculation for N^{-} part of diffraction coefficient

where:

$$\beta = \begin{Bmatrix} \phi - \phi', \text{ for the incident case} \\ \phi + \phi', \text{ for the reflection case} \end{Bmatrix}, \text{ (denoted by BET)}$$

$$a^{\pm}(\beta) = 2\cos^2 \underbrace{\left(\frac{2n\pi N^{\pm} - \beta}{2}\right)}_{ACS}, \text{ (denoted by A).}$$

$N^{\pm}$ are the integers which most nearly satisfy (by truncating the floating point values for $N^{\pm}$) the equations:

$$2\pi n N^{+} - \beta = \pi$$
$$2\pi n N^{-} - \beta = -\pi$$

F(x) is the transition function. It is contained mostly in FA with a constant contained in COTA.

n is the wedge number FN.

For the corner diffraction term (LOG = .TRUE.), the coefficient is given as (see reference B):

$$\underbrace{DICOEF(R,\beta,\sin\beta_o,n,R_c)}_{DIR} = \underbrace{\frac{-e^{-j\pi/4}}{2n\ 2\pi k\ \sin\beta_o}}_{\frac{TOP}{DEM} = COM} \Bigg\{ \underbrace{\cot\left(\frac{\pi+\beta}{2n}\right) F\left[kRa^{+}(\beta)\right]}_{COTA*FA} \times \left| F\left[\frac{Ra^{+}(\beta)/\lambda}{kR_c a(\pi+\beta_o-\beta_c)}\right]\right| +$$

DIR

TOP/DEM = COM

COTA*FA

BABS(FFCT(R*A*DELU)), where BABS is a function subprogram to give the absolute value of the argument, and FFCT is a function subprogram to return the transition function (See BABS and FFCT)

Used in UPPI calculation for N^+ part of diffraction coefficient

$$\underbrace{\cot\left(\frac{\pi-\beta}{2n}\right) F\left[kRa^{-}(\beta)\right]}_{COTA*FA} \left| F\left[\frac{Ra^{-}(\beta)/\lambda}{kR_c a(\pi+\beta_o-\beta_c)}\right]\right| \Bigg\},$$

COTA*FA

BABS(FFCT(R*A*DELU)), where BABS is a function subprogram to give the absolute value of the argument, and FFCT is a function subprogram to return the transition function (See BABS and FFCT)

Used in UNPI calculation for N^- part of diffraction coefficient

where R_c is the corner distance parameter and β_c is the polar angle measured from the corner. R_c and β_c are contained in DELU which is computed from DELL, an input formal parameter. An illustration of the geometry is given in figure 1.

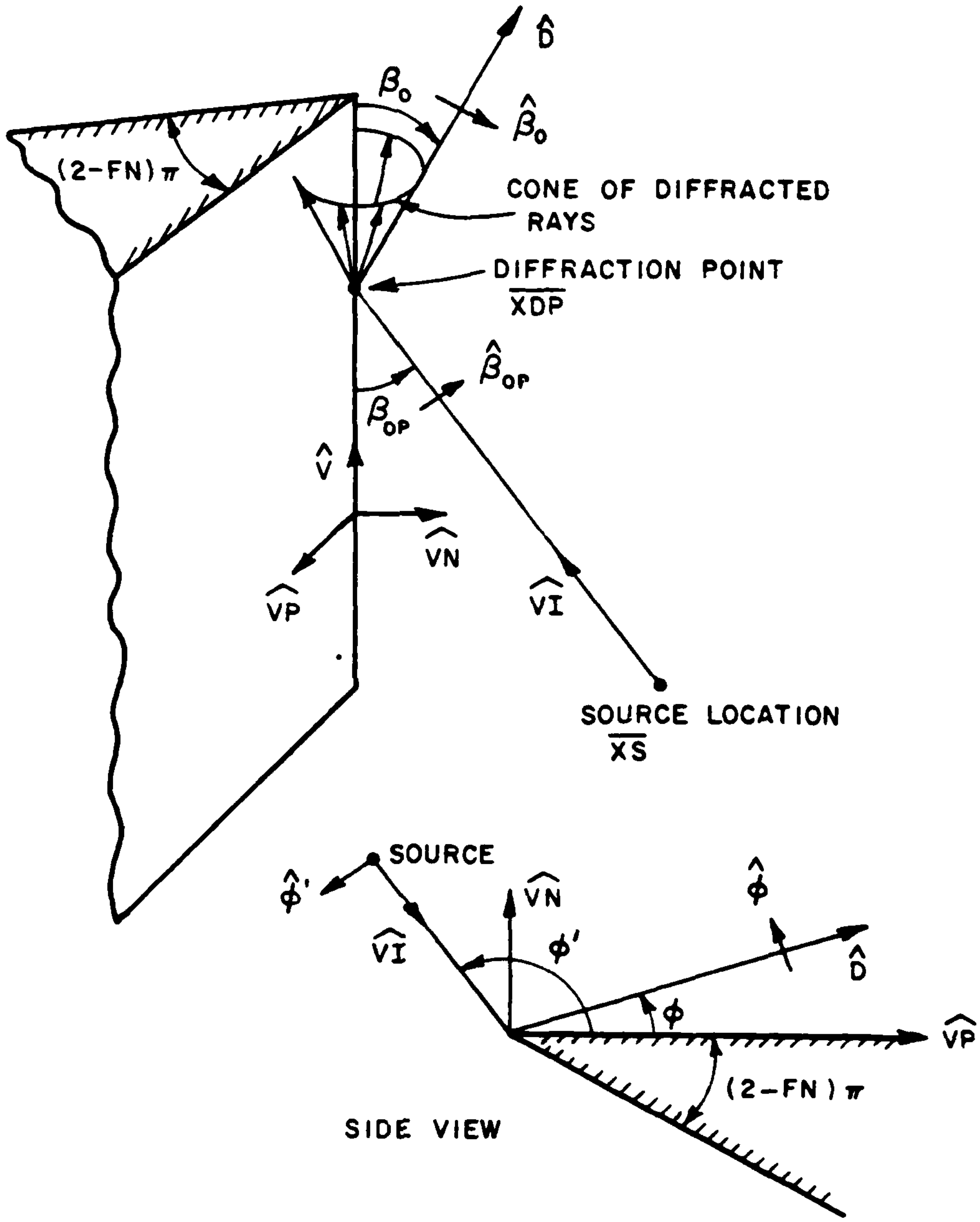


Figure 1. Edge Diffraction Geometry

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
A	Angular function for transition function
ACS	Computational variable in calculation of A
ANG	BET in radians
BET	Angular argument of diffraction coefficient (β, PH+PHP, or PH-PHP) in degrees
BOTL	Argument of transition function
C	Real part of Fresnel integral
COM	Constant for diffraction coefficient
COTA	Cotangent times the square root of the A function
DEL	Corner part of argument for the corner transition function correction term
DELL	Part of argument for the corner transition function correction term
DELU	Inverse of DEL
DEM	4*PI*FN*SIN(BO)
DIR	Incident or reflection part of diffraction coefficient
DN	Integer which most nearly satisfies the equation, 2*PI*FN*DN-BET=PI OR -PI
DNS	Computational variable
EX	CEXP (J*K*R*A)
FA	Transition function without $\sqrt{A}$
FN	Wedge angle number
LOG	A logical variable set true if the corner diffraction coefficient is to be computed

N	Computational variable
PI	π
R	Distance parameter which is a function of source type, incidence angle, source-to-diffraction point separation distance
RAG	Argument of cotangent term
RPD	The conversion factor for converting angular measurements in degrees to radians (= PI/180. = 0.0174532925)
S	Imaginary part of Fresnel integral
SBO	Sine of β_o
SGN	Sign of DNS
SQR	$\sqrt{(2*PI*R)}$
TOP	The complex constant -CEXP(-J*PI/4).
TPI	2π
TS	Absolute value of TSIN
TSIN	Sine of argument of cotangent term
UNPI	N- component of DIR
UPPI	N+ component of DIR

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
BET	F.P.
DELL	F.P.
FN	F.P.
LOG	F.P.
PI	/PIS/

R	F.P.
RPD	/PIS/
SBO	F.P.
TOP	/TOPD/
TPI	/PIS/

B. OUTPUT LOCATION

DIR	F.P.

6. CALLING ROUTINES:

DIFPLT

DPLRPL

DW

RPLDPL

7. CALLED ROUTINES:

BABS

BEXP

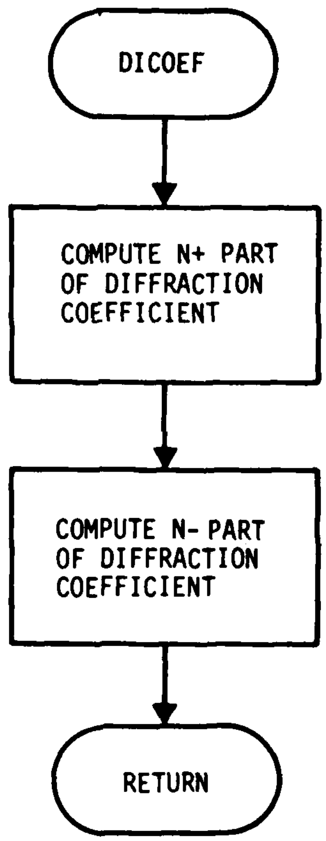
FFCT

FRNELS

8. REFERENCES:

A. R. G. Kouyoumjian and P. H. Pathak, "A Uniform Geometrical Theory of Diffraction for an Edge in a Perfectly Conducting Surface," Proc. IEEE, Vol. 62, November 1974, pp 1448-1461.

B. W. D. Burnside and P. H. Pathak, "A Corner Diffraction Coefficient," to appear.


DICOEF
COMPUTE N+ PART OF DIFFRACTION COEFFICIENT
COMPUTE N- PART OF DIFFRACTION COEFFICIENT
RETURN

1. NAME: DIFPLT (GTD)

2. PURPOSE: To determine the unobstructed diffracted field from edge ME of plate MP from a unit source. Slope diffracted fields are included if the user requested them. Corner diffracted fields are included for far-field calculations only if the corner diffraction was requested.

3. METHOD: DIFPLT is the driver routine which directs all the ray tracing, physics and field calculations for the diffracted field off edge ME of plate MP from a unit source in the given far-field direction or to a given near-field observation point. The slope and corner diffraction terms are included also. The related geometry is shown in figure 1. A detailed explanation of the fields calculated is given in reference A. The code first initializes the fields to zero, then determines where the diffraction point is by calling subroutine DFPTWD. If diffraction does not occur, debug information (if requested) is printed on file LUPRNT and control returns to the calling routine. If a diffraction point is found on the tangent to the plate edge, but it is not within the corners, it is set at the closest corner (see figure 2). Then the ray path is checked for obstructions by first checking the ray from the diffraction point in the far-field observation direction or to the near-field observation point, and then checking the ray from the source to the diffraction point. If either path is obstructed, the code checks to see if diffractions for this edge have been found before and, if so, identifies the edge as possibly double diffracting. Then a flag is set to indicate a diffracted field was not found this time. Debug information is printed (if requested) and control then returns to the calling routine. If the ray path is clear of objects, then diffraction angles from different orientations, polarization unit vectors, and other related geometry are calculated.

The distance parameter (TPP) needed for the edge diffracted field to be continuous at the shadow and reflection boundaries is given as:

$$TPP = S' \sin(\beta_o)^2, \text{ for far field}$$

and

$$TPP = \frac{S\, S' \sin(\beta_o)^2}{S + S'}, \text{ for near field}$$

where

S' = SP = distance from source to diffraction point
S = SNF = distance from diffraction point to observation point
$\sin(\beta_o)$ = SBO = sine of the diffraction angle (the angle the diffracted ray makes with the edge).

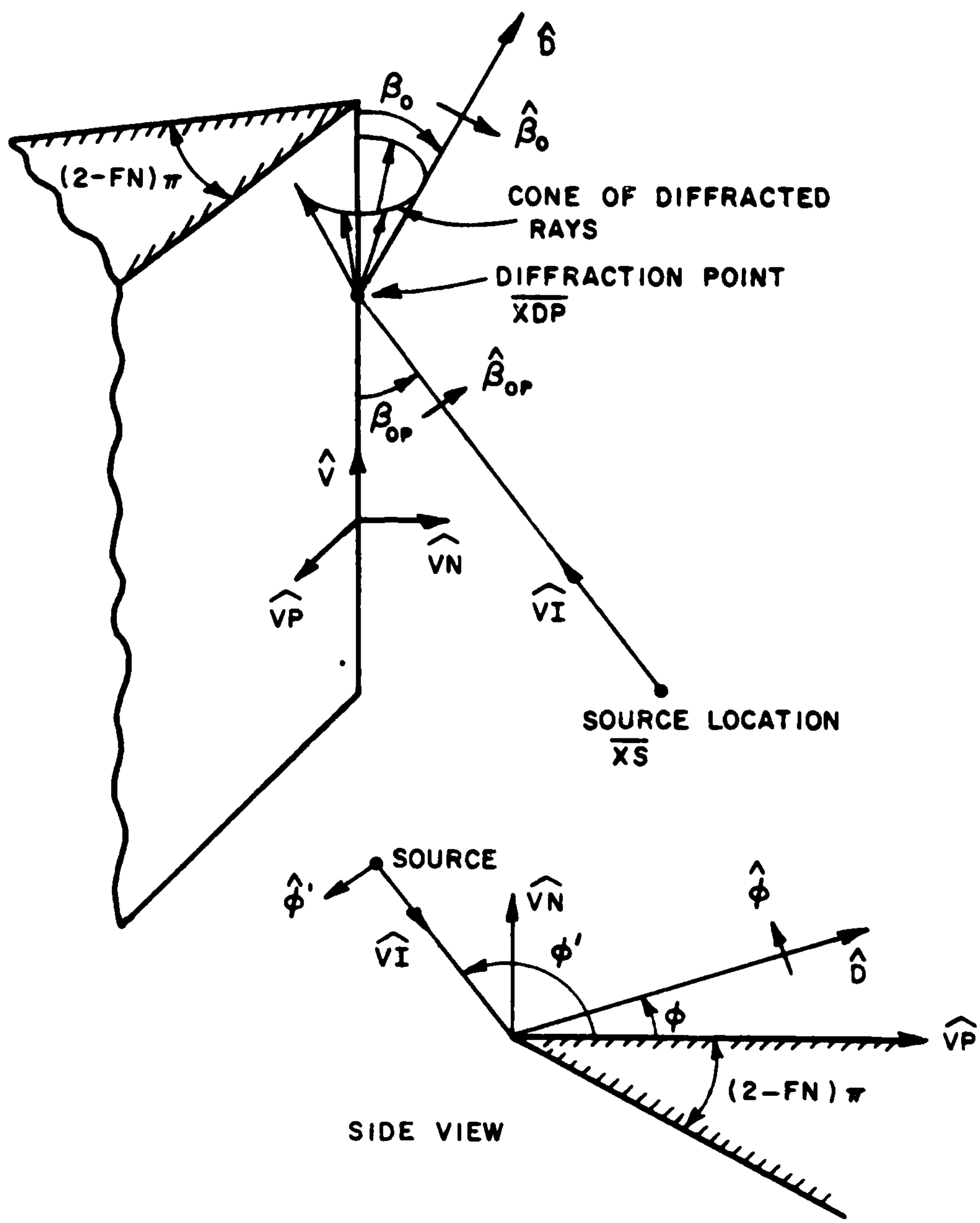


Figure 1. Edge Diffraction Geometry

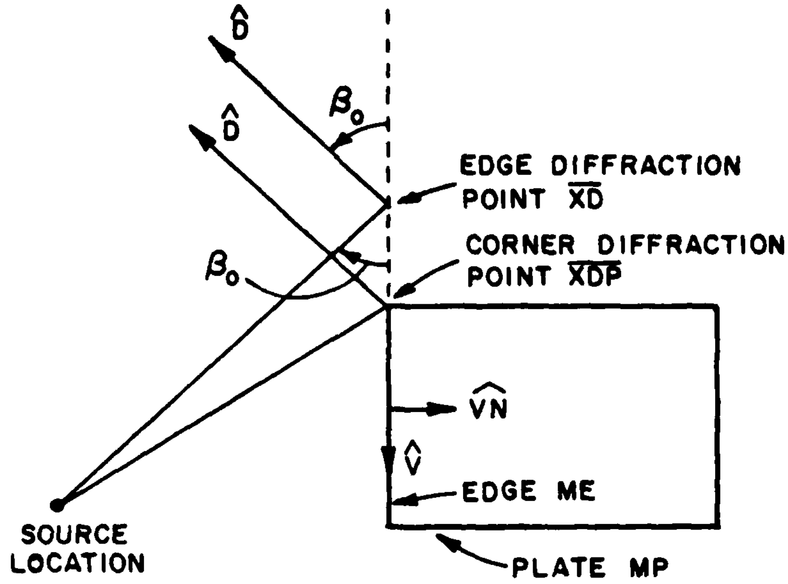


Figure 2. Far-field Corner Diffraction Geometry

(The distance parameter is needed in the diffraction coefficient subroutines DW and DICOEF.) The source field pattern factor, based on the source location, is found by calling subroutine SOURCE. If slope diffraction was requested, the incident slope field pattern factor is found by calling subroutine SOURCP. Then the phase factor is computed; it refers a far-field electric field back to the reference coordinate system (RCS) origin, and for near-field calculations it includes the spherical wave spread factor.

The edge diffraction coefficients for the hard and soft boundaries are computed by calling subroutine DW. Then the perpendicular and parallel components of the diffracted field are computed by multiplying the source field pattern factor, the phase factor, and the diffraction coefficients. If slope diffraction was requested, its field is added. The field is converted to theta and phi components in the RCS and is then converted to x, y, z components and accumulated in subroutine XYZFLD.

If corner diffracted fields were not requested, debug information (if requested) is printed on file LUPRNT giving the diffracted field magnitude and theta and phi components. Control is then returned to the calling routine.

If corner diffraction was requested, far-field diffracted fields only are computed for one corner on edge ME and then for the other corner. Subroutine DICOEF is called for the corner diffraction coefficients. The fields are calculated in parallel and perpendicular components and then converted to theta and phi components in the RCS. The field is then converted to x, y, z components and added to the fields already accumulated by calling subroutine XYZFLD.

The code ends with printing debug information (if requested) on file LUPRNT.

4. INTERNAL VARIABLES

VARIABLE	DEFINITION
ADN	Dot product of vector from plate MP to the source and the plate unit normal
AFN	Wedge angle number
BETN	Difference in diffracted and incident phi angles
BETP	Sum of diffracted and incident phi angles
BO	Diffracted field beta polarization unit vector (in edge-fixed coordinate system) in RCS components
BOP	Incident field beta polarization unit vector (in edge-fixed coordinate system) in RCS components
CNP	Cosine of half wedge angle
CORN	Corner diffraction coefficient
CPH	Cosine of PSR
CPHO	Cosine of PSOR
CTH	Cosine of THR
CTHP	Cosine of THPR
DEL	Parameter used in transition function
DH	Diffraction coefficient for hard boundary condition
DHIT	Distance from source to nearest hit point (from subroutine PLAINT or CYLINT)

DPH	Slope diffraction coefficient for hard boundary condition
DPR	Degrees per radian conversion factor
DPS	Slope diffraction coefficient for soft boundary condition
DS	Diffraction coefficient for soft boundary condition
DV	Dot product of edge vector and propagation direction unit vector, D, which is the cosine of beta
ECBI	Edge diffraction coefficient (from subroutine DICOEF) for incident diffracted field modified for corner diffraction
ECBR	Edge diffraction coefficient (from subroutine DICOEF) for reflected diffracted field modified for corner diffraction
ECPH	Phi component of corner diffracted E-field
ECTH	Theta component of corner diffracted E-field
EDPH	Phi component of edge diffracted E-field
EDPL	Component of diffracted field parallel to the edge
EDPR	Component of diffracted field perpendicular to the edge
EDTH	Theta component of edge diffracted E-field
EF	Theta component of corner diffracted field in RCS
EG	Phi component of corner diffracted field in RCS
EIPL	Component of incident field parallel to the edge
EIPLP	Pattern factor for component of source (incident) slope field parallel to the edge

EIPR	Component of incident field parallel to the edge
EIPRP	Pattern factor for component of the source (incident) slope field perpendicular to the edge
EIX,EIY,EIZ	Source pattern factors for x,y, and z components of incident E-field
EXPH	Complex phase term (refer phase to RCS origin)
FN	Wedge angle number
FNN	Wedge angle indicator
FNP	Angle exterior to wedge in degrees
GAM	Dot product of the propagation direction and the vector from the origin to the diffraction point
IDD	Double diffraction shadow boundary identification array
ISN	Sign change variable
LCORNR	Logical variable set true if corner diffraction is requested
LDIF	Logical variable set false if the edge tangent diffraction point does not lie on the edge between the two corners. (If this happens the diffraction point is moved to the nearest corner and only corner diffraction, if requested, is computed.)
LDIFFR	Logical variable set true if edge diffraction occurred
LHIT	Set true if ray hits a plate or cylinder (from subroutine PLAINT or CYLINT)
LSLOPE	Logical variable set true if slope diffraction is requested
MC	Corner at end of edge ME
ME	Edge on plate MP where diffraction occurs

MP	Plate for which diffraction occurs
N	DO loop variable
PD	Dot product of edge binormal and diffracted ray propagation direction
PH	Diffracted field phi polarization unit vector (in edge-fixed coordinate system) in RCS components
PHIR	Phi component of incident ray propagation direction in RCS
PHO	Incident field phi polarization unit vector (in edge-fixed coordinate system) in RCS components
PHSR	Phi component of diffracted ray propagation direction in RCS
PP	Negative dot product of edge binormal and incident ray propagation direction
PS	PSR*DPR
PSD	Diffracted ray phi angle in edge-fixed coordinate system
PSO	PSOR*DPR
PSOD	Incident ray phi angle in edge-fixed coordinate system
PSOR	Phi component of incident ray direction in edge-fixed coordinate system
PSR	Phi component of diffracted ray propagation direction in edge-fixed coordinate system
QD	Dot product of plate normal and diffracted ray propagation direction
QI	Negative of dot product of plate normal and incident ray propagation direction
RM	Magnitude of vector from corner ME to source

RX,RY,RZ	X,Y, and Z components of vector from corner MC to source
SBO	Sine of BO, the angle the diffracted ray makes with the edge unit vector
SNF	Distance between near-field observation point and diffraction point
SNP	Sine of half wedge angle
SP	Distance from source to diffraction point (from subroutine DFPTWD)
SPH	Sine of PSR
SPHO	Sine of PSOR
SPP	Distance from source to diffraction point
STHR	Sine of THR
TERM	Coefficient of corner diffracted fields
THIR	Theta component of incident ray direction in reference coordinate system
THPR	Angle diffracted ray makes with edge
THR	Angle between edge unit vector and ray from source to corner MC
TPP	Distance parameter used in calculating diffraction coefficients
VECT	Vector used to move diffraction point off edge for shadowing tests
VI	Unit vector of incident ray propagation direction (from subroutine DFPTWD)
VIP	Unit vector from source to diffraction point
VMG	Distance along the edge from first corner of edge to diffraction point
VXS	3 x 3 matrix defining the source coordinate system axes

XC	Corner location for corner diffraction
XD	Diffraction point (calculated in subroutine DFPTWD)
XDP	Diffraction point (used for shadowing tests)
XS	Source location in RCS
XSS	Source location
ZP	Dot product of diffracted ray propagation direction unit vector $\hat{D}$ and vector from diffraction point to corner MC

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
D	/DIR/
DP	/THPHUV/
DPR	/PIS/
DT	/THPHUV/
FLDPT	/NEAR/
FNN	F.P.
IANG	/DOUBLE/
ID	/DOUBLE/
IDD	/DOUBLE/
LCORNR	/LOGDIF/
LDEBUG	/TEST/
LNRFLD	/NEAR/
LSLOPE	/LOGDIF/
LSURF	/SURFAC/
LUPRNT	/ADEBUG/

	ME	F.P.
	MEP	/GEOPLA/
	MP	F.P.
	MPH	/HITPLT/
	PHSR	/DIR/
	PI	/PIS/
	THSR	/DIR/
	TPI	/PIS/
	V	/GEOPLA/
	VMAG	/EDMAG/
	VN	/GEOPLA/
	VP	/GEOPLA/
	VXS	/SORINF/
	X	/GEOPLA/
	XS	/SORINF/
B.	OUTPUT	LOCATION
	ECPH	F.P.
	ECTH	F.P.
	EDPH	F.P.
	EDTH	F.P.

6. CALLING ROUTINE:

GTDDRV

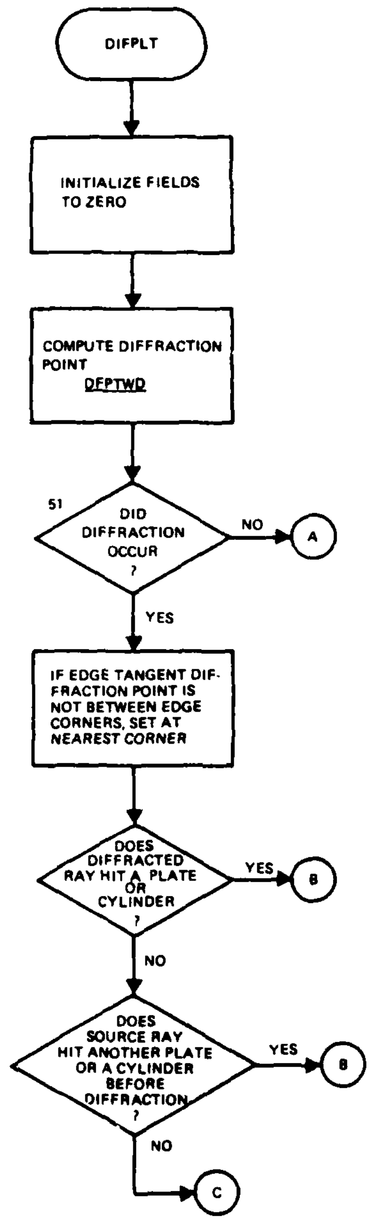
7. CALLED ROUTINE:

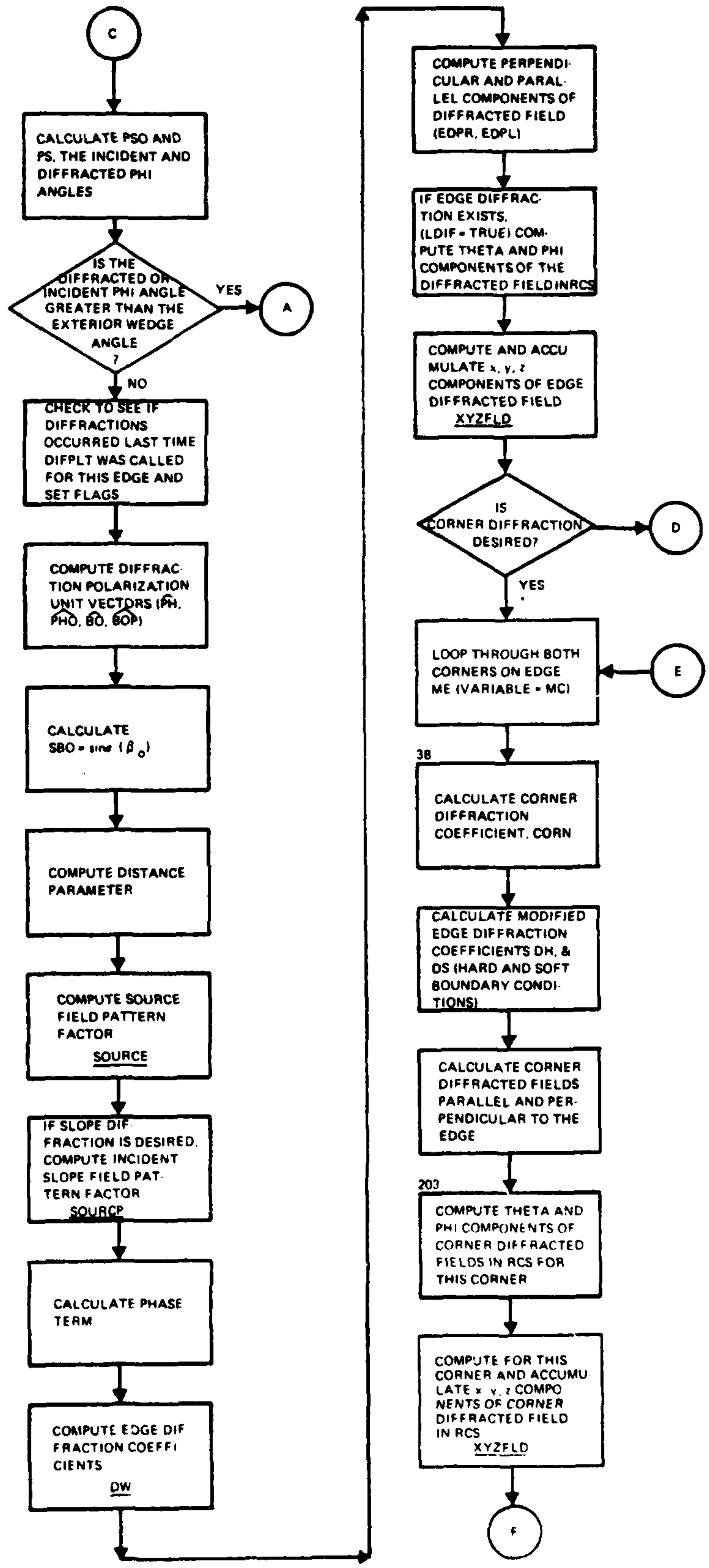
ASSIGN	PLAINT
BEXP	SMAGNF

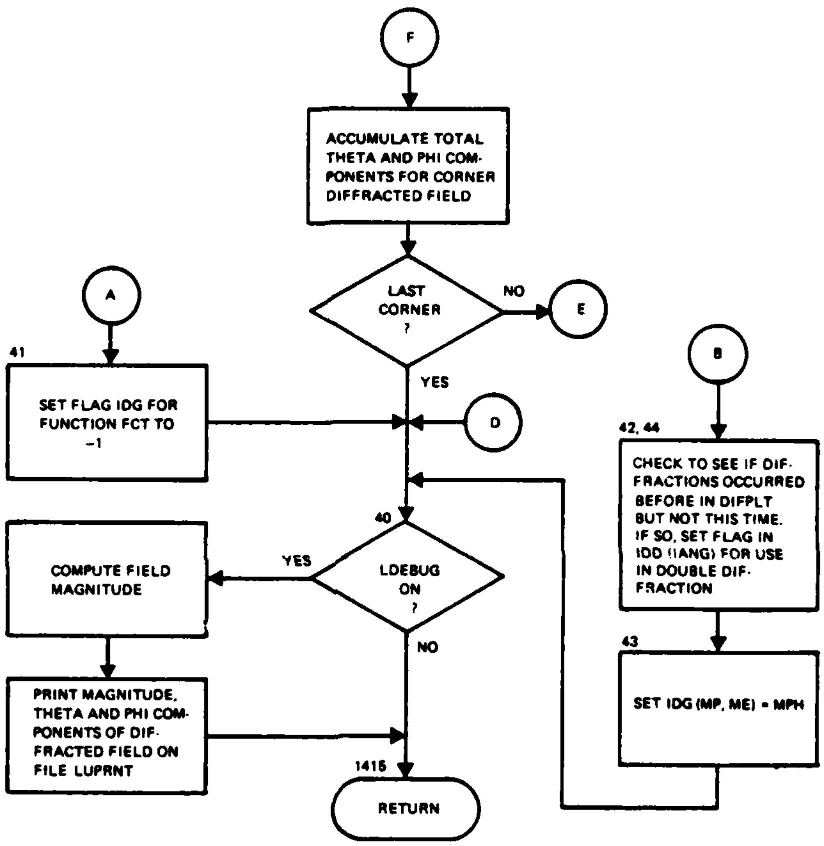
BTAN2	SOURCE
CYLINT	SOURCP
DFPTWD	STATIN
DICOEF	STATOT
DW	TPNFLD
FFCT	WLKBCK
NFD	XYZFLD

8. REFERENCE:

A. R. G. Kouyoumjian and P. H. Pathak, "A Uniform Geometrical Theory of Diffraction for an Edge in a Perfectly Conducting Surface," Proc. IEEE, Vol. 62, November 1974, pp. 1448-1461.


DIFPLT
INITIALIZE FIELDS TO ZERO
COMPUTE DIFFRACTION POINT
DFPTWD
51
DID DIFFRACTION OCCUR ?
NO
A
YES
IF EDGE TANGENT DIF-FRACTION POINT IS NOT BETWEEN EDGE CORNERS, SET AT NEAREST CORNER
DOES DIFFRACTED RAY HIT A PLATE OR CYLINDER ?
YES
B
NO
DOES SOURCE RAY HIT ANOTHER PLATE OR A CYLINDER BEFORE DIFFRACTION ?
YES
B
NO
C


C
CALCULATE PSO AND PS, THE INCIDENT AND DIFFRACTED PHI ANGLES
IS THE DIFFRACTED OR INCIDENT PHI ANGLE GREATER THAN THE EXTERIOR WEDGE ANGLE ?
YES
A
NO
CHECK TO SEE IF DIFFRACTIONS OCCURRED LAST TIME DIFPLT WAS CALLED FOR THIS EDGE AND SET FLAGS
COMPUTE DIFFRACTION POLARIZATION UNIT VECTORS (PH, PHO, BO, BOP)
CALCULATE SBO = sine (β o)
COMPUTE DISTANCE PARAMETER
COMPUTE SOURCE FIELD PATTERN FACTOR SOURCE
IF SLOPE DIFFRACTION IS DESIRED, COMPUTE INCIDENT SLOPE FIELD PATTERN FACTOR SOURCP
CALCULATE PHASE TERM
COMPUTE EDGE DIFFRACTION COEFFICIENTS DW
COMPUTE PERPENDICULAR AND PARALLEL COMPONENTS OF DIFFRACTED FIELD (EDPR, EDPL)
IF EDGE DIFFRACTION EXISTS, (LDIF = TRUE) COMPUTE THETA AND PHI COMPONENTS OF THE DIFFRACTED FIELD IN RCS
COMPUTE AND ACCUMULATE x, y, z COMPONENTS OF EDGE DIFFRACTED FIELD XYZFLD
IS CORNER DIFFRACTION DESIRED?
D
YES
LOOP THROUGH BOTH CORNERS ON EDGE ME (VARIABLE = MC)
E
38
CALCULATE CORNER DIFFRACTION COEFFICIENT, CORN
CALCULATE MODIFIED EDGE DIFFRACTION COEFFICIENTS DH, & DS (HARD AND SOFT BOUNDARY CONDITIONS)
CALCULATE CORNER DIFFRACTED FIELDS PARALLEL AND PERPENDICULAR TO THE EDGE
203
COMPUTE THETA AND PHI COMPONENTS OF CORNER DIFFRACTED FIELDS IN RCS FOR THIS CORNER
COMPUTE FOR THIS CORNER AND ACCUMULATE x, y, z COMPONENTS OF CORNER DIFFRACTED FIELD IN RCS XYZFLD
F


F
ACCUMULATE TOTAL THETA AND PHI COMPONENTS FOR CORNER DIFFRACTED FIELD
LAST CORNER ?
NO
E
YES
A
41
SET FLAG IDG FOR FUNCTION FCT TO −1
D
B
42, 44
CHECK TO SEE IF DIFFRACTIONS OCCURRED BEFORE IN DIFPLT BUT NOT THIS TIME. IF SO, SET FLAG IN IDD (IANG) FOR USE IN DOUBLE DIFFRACTION
40
LDEBUG ON ?
YES
COMPUTE FIELD MAGNITUDE
NO
43
SET IDG (MP, ME) = MPH
PRINT MAGNITUDE, THETA AND PHI COMPONENTS OF DIFFRACTED FIELD ON FILE LUPRNT
1415
RETURN

1. NAME: DMPDRV (GTD, INPUT, MOM, OUTPUT)

2. PURPOSE: Execute an expression.

3. METHOD: DMPDRV will execute simple expressions with no precedence on the operators. Precedence can be obtained by using parentheses. The argument list must be operand, operator, operand for all operations.

   Example 1: FRQ = 300. (operand1 operator operand2)
   The first operator must be an equal sign. The operator precedence is from right to left for all operations.

   Example 2: FRQ = N*2 + 10
   The 10 is added to the 2 before the N is multiplied. Precedence can be obtained by doing the following:

   Example 3: FRQ = (N*2) + 10
   DMPDRV uses right to left precedence on all operations starting with the innermost parentheses.

4. INTERNAL VARIABLE:

VARIABLE	DEFINITION
ADDOPR	Logical operator flag for ADD
C	Resultant of one of four operations
CMAG	Magnitude of C
CMPLX1	Logical flag of first symbol
CMPLX2	Logical flag of second symbol
COP1	Complex operand one
COP2	Complex operand two
DIVOPR	Logical operator flag for division
EXPOPR	Logical operator flag for exponentiation
IBITR	Attribute bit for a complex number
IBIT1	Attribute word for the first symbol
IBIT2	Attribute word for the second symbol
ILP	Index to left parenthesis

DMPDRV (GTD, INPUT, MOM, OUTPUT)

IOPR	Operator for matrix operations
IPAREN	Parentheses counter
IRP	Index to right parenthesis
LITTYP	Type of argument in the literal table
LOCARG	Field pointer
LOCLIT	Pointer to the literal table
MATOP1	Logical flag for matrix operand one
MATOP2	Logical flag for matrix operand two
MULOPR	Logical operator flag for multiplication
NAMFIL	File name where the resultant matrix is stored
NAMOPR	Operand name
NAMOP1	Name of operand one
NAMOP2	Name of operand two
NAMSYM	Symbol name
NCOL1	Number of columns in operand one
NCOL2	Number of columns in operand two
NDXARG	Pointer to the literal table
NDXKYW	Index to keyword array
NROW1	Number of rows in operand one
NROW2	Number of rows in operand two
NXTARG	Pointer to the first operand after the equal sign
R	Location of real operation resultant
ROP1	Real operand one
ROP2	Real operand two
SUBOPR	Logical operator flag for subtraction

5. I/O VARIABLES:

A. INPUT	LOCATION
FLTLIT	/PARTAB/
INTARG	/ARGCOM/
NDATBL	/PARTAB/
NPDATA	/PARTAB/
NUMARG	/ARGCOM/

B. OUTPUT	LOCATION
NDATBL	/PARTAB/

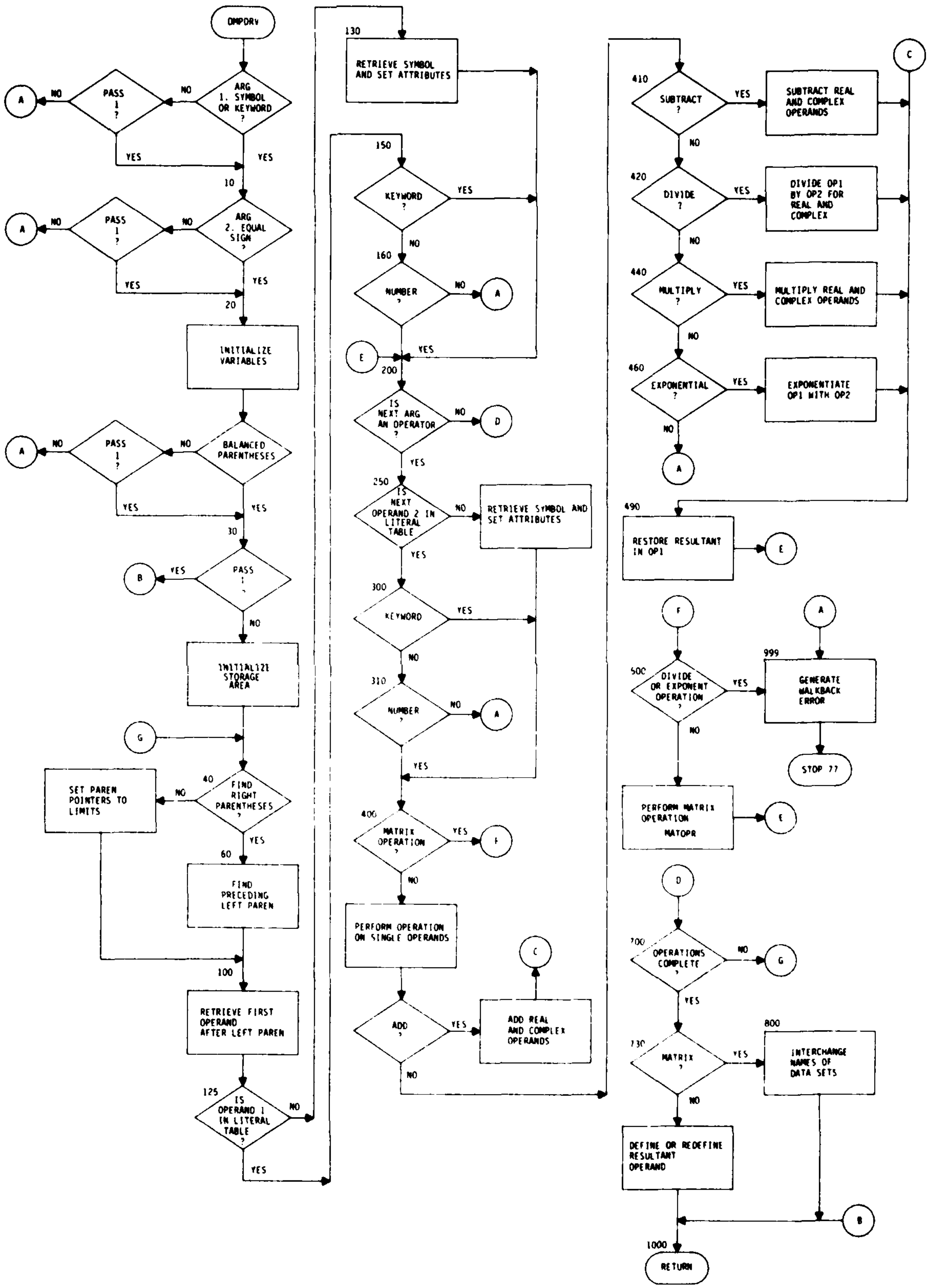
6. CALLING ROUTINE:

TSKXQT

7. CALLED ROUTINES:

ASSIGN	PUTKWV
CLSFIL	PUTSYM
CONVRT	STATIN
ERROR	STATOT
GETKWV	SYMDEF
GETSYM	WLKBCK
IBITCK	ZZXDUM

DMPDRV (GTD, INPUT, MOM, OUTPUT)



1. NAME: DPI (GTD)

2. PURPOSE: To calculate the incident part or the reflection part of the wedge slope diffraction coefficient.

3. METHOD: This subroutine computes either the incident part or the reflection part of the wedge slope diffraction coefficient depending upon the value of BET. This coefficient is based upon the Uniform Geometrical Theory of Diffraction (see reference A and equation 15 of reference B). The geometry associated with the calculation of the coefficient is shown in figure 1. The slope diffraction coefficient is given as:

$$DPI(R,\beta,\sin\beta_o,n) = \frac{-e^{-j\pi/4}}{4n^2\sqrt{2\pi k}\,\sin\beta_o}\left\{\csc^2\left(\frac{\pi+\beta}{2n}\right)F_s[kRa^+(\beta)] - \csc^2\left(\frac{\pi-\beta}{2n}\right)F_s[kRa^-(\beta)]\right\}$$

DPIR	COM = $\frac{-TOP}{DEM}$	CSCA*FPA Used in UPPI Calculation for N^+ Part of Coefficient	CSCA*FPA Used in UNPI Calculation for N^- Part of Coefficient

$$\text{where } \beta = \begin{cases} \phi - \phi', & \text{for the incident case} \\ \phi + \phi', & \text{for the reflection case} \end{cases}, \text{ (denoted by BET),}$$

$$a^{\pm}(\beta) = 2\cos^2\left(\frac{2n\pi N^{\pm} - \beta}{2}\right), \quad \text{(denoted by A)}$$

$N^{\pm}$ are the integers which most nearly satisfy (by truncating the floating point values for $N^{\pm}$) the equations:

$$2\pi n N^+ - \beta = \pi$$

$$2\pi n N^- - \beta = -\pi.$$

$F_s(x) = 2jx\,[1 - F(x)]$ is the transition function. It is contained in FPA except for a constant which is included in the calculation for CSCA. The wedge angle number n is denoted by FN.

DPI (GTD)

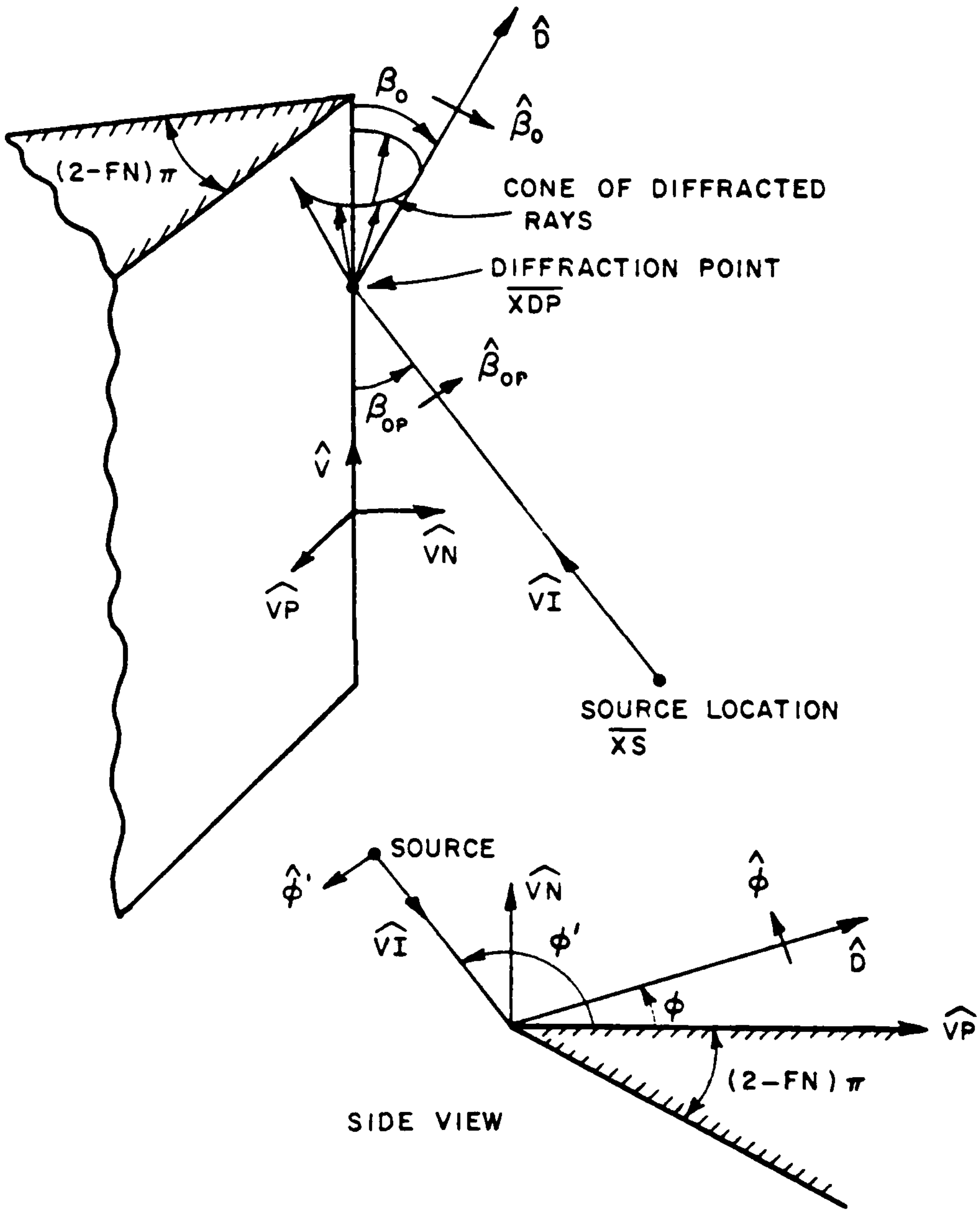


Figure 1. Edge Diffraction Geometry

The negative value of TOP, which is contrary to theory, is compensated for in subroutine SOURCP.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
A	Angular function for transition function
ANG	BET in radians
BET	Angular argument of diffraction coefficent (B, PH+PHP or PH-PHP)
BOTL	Argument of transition function
C	Real part of Fresnel integral
COM	Constant for slope diffraction coefficient
CSCA	Cosecant times the A function
DEM	8*PI*FN*FN*SIN(BO)
DN	Integer which most nearly satisfies the equation, 2*PI*FN*DN-BET=PI or -PI
DNS	Computational variable
DPIR	Incident or reflection part of slope diffraction coefficient
EX	CEXP(J*K*R*A)
FN	Wedge angle number
FPA	Slope transition function without the A function

PI	π
N	Computational variable
R	Distance parameter which is a function of source type, incidence angle, and source-to-diffraction point separation distance and is such that the field is continuous at shadow and reflection boundaries
RAG	Argument of cosecant term
RPD	Radians per degree, π/180
S	Imaginary part of Fresnel integral
SBO	Sine of β
SGN	Sign of DNS
TOP	The complex constant, -CEXP(-J*PI/4)
TPI	2π
TS	TSIN squared
TSIN	Sine of argument of cosecant term
UNPI	N- component of DPIR
UPPI	N+ component of DPIR

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
BET	F.P.
FN	F.P.
PI	/PIS/
R	F.P.
RPD	/PIS/

SBO	F.P.
TOP	/TOPD/
TPI	/PIS/
B. OUTPUT	LOCATION
DPIR	F.P.

6. CALLING ROUTINE:

   DW

7. CALLED ROUTINES:

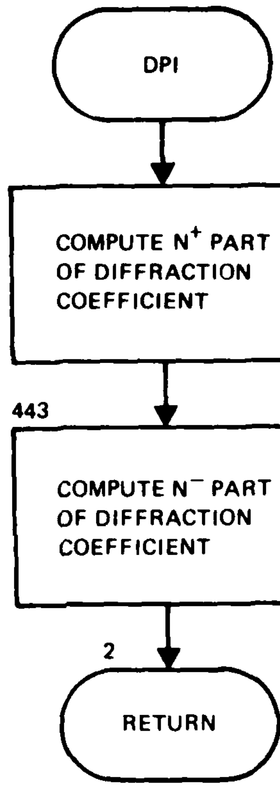
   BEXP

   FRNELS

8. REFERENCES:

A. Y. M. Hwang and R. G. Kouyoumjian, "A Dyadic Diffraction Coefficient for an Electromagnetic Wave Which is Rapidly Varying at an Edge," USNC-URSI 1974 Annual Meeting, Boulder, CO., Oct. 1974.

B. R. G. Kouyoumjian and P. H. Pathak, "A Uniform Geometrical Theory of Diffraction for an Edge in a Perfectly Conducting Surface," Proc. IEEE, Vol. 62, November 1974, pp. 1448-1461.


DPI
COMPUTE N+ PART OF DIFFRACTION COEFFICIENT
443
COMPUTE N− PART OF DIFFRACTION COEFFICIENT
2
RETURN

1. NAME: DPLRCL (GTD)

2. PURPOSE: To compute the unobstructed electric field from a unit source ray diffracted by edge ME of plate MP and then reflected by a cylinder in the given far-field observation direction or to a given near-field observation point.

3. METHOD: DPLRCL is the driver routine which directs all the ray tracing, physics and field calculations for determining the electric field diffracted by a plate edge and reflected from a cylinder in the given far-field observation direction or to the given near-field observation point. The field diffracted by the plate edge is found using the Uniform Geometrical Theory of Diffraction. (See reference A). This causes an astigmatic tube of rays to be incident on the cylinder. The field reflected by the cylinder is found using geometrical optics (see reference A). Pertinent geometry is shown in figure 1.

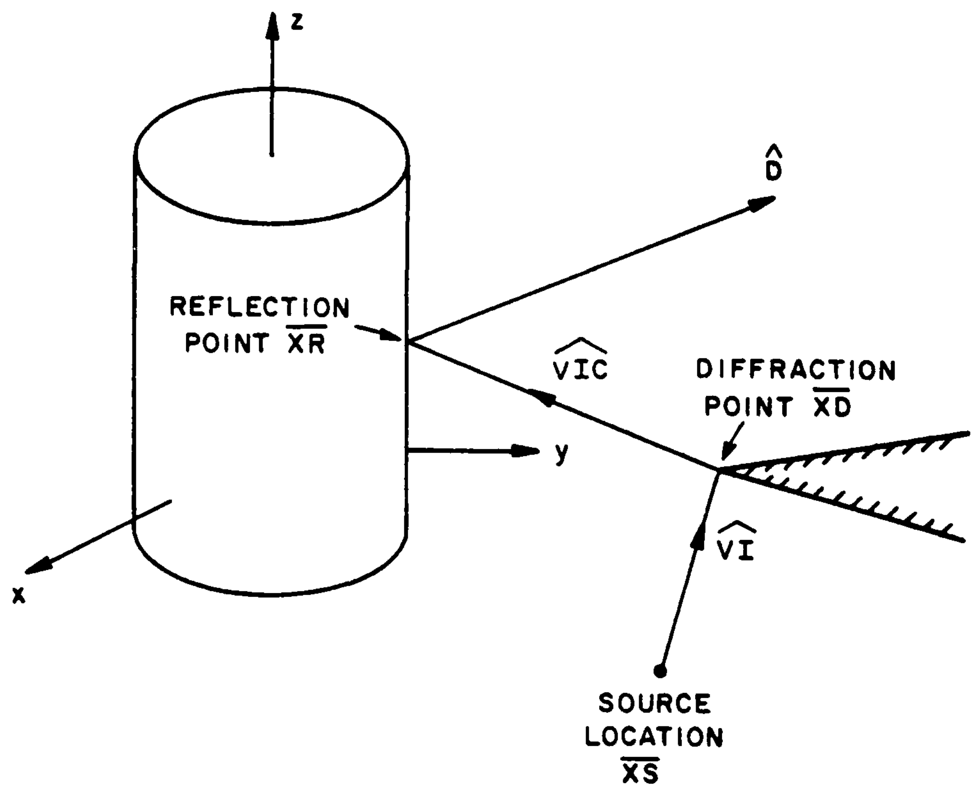


Figure 1. Illus‿ration of a Ray Diffracted by a Plate Edge and then Reflected by the Cylinder

The code first checks to see if edge ME of plate MP is formed by intersecting plates. If it is, the fields are set to zero. Debug information (if requested) is computed and printed on file LUPRNT. Control is then returned to the calling routine. If edge ME is not on intersecting plates, the code determines the ray path for the plate diffracted-cylinder reflected field. This is done differently depending upon whether far field or near field was requested: if far field was requested, the code makes a quick check to see if diffraction is possible. If it is not, a flag is set to indicate that no starting data are available for the next time DPLRCL is called. The field is set to zero. Debug information is printed if it was requested, and control is returned to the calling routine. If it is possible for diffraction to occur, subroutine DFRFPT is called to compute the diffracted-reflected ray path. For near field, the code interchanges the source and observation point locations. Subroutine RFDFPT is called to compute the ray path which would be reflected from a cylinder and then diffracted by a plate. After the ray path has been found, subroutine DPLRCL switches the source and observation point locations back to their original positions. The unit direction vectors are also negated so that the proper direction for a plate diffracted-cylinder reflected field can be found. Now for both near field and far field the code checks to make sure that reflection and diffraction did occur. If they did not, the fields are set to zero and debug information is printed (if requested). Control is returned to the calling routine. If the diffraction and reflection are legal (by satisfying Snell's Law), the code then checks the total ray path for any obstructions. If it is obstructed, the fields are set to zero, and debug information is printed. Control is returned to the calling routine. If the ray path is clear, field computations can begin.

The total diffracted-reflected field is found by first determining the fields incident upon the diffraction point. The source field pattern factor is found by calling subroutine SOURCE and the diffracted field is found. The diffraction coefficient is determined by calling subroutine DW. The phase factor and ray spreading radii for the field incident on the cylinder are computed and combined with the diffraction coefficient and the field incident upon the plate to determine the diffracted field. Geometrical optics is used to determine the reflection parameters and reflected field. A phase factor and ray spreading factors for the diffracted-reflected field are found and combined with the field incident upon the cylinder to determine the total diffracted-reflected field. This field is given in theta and phi components. Subroutine XYZFLD is called to compute the x, y, z components of the total field and to accumulate this field with other fields from previous interactions. The total diffracted-reflected field has the form shown on pages 163 and 164 of reference B.

If debug information is requested, the field magnitude is computed. The field magnitude, theta and phi components are printed on file LUPRNT. Control is then returned to the calling routine.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
BO	Diffracted field polarization unit vector parallel to edge
BOP	Incident field polarization unit vector parallel to edge
DD	Distance from z axis to cylinder reflection point in the x-y plane
DD1	Dot product of source ray diffracted from plate tangent to tangent point 1 of cylinder and propagation direction (2-D)
DD2	Dot product of source ray diffracted from plate tangent to tangent point 2 of cylinder and propagation direction (2-D)
DH	Diffraction coefficient for hard boundary condition
DHIT	Distance to hit point on plate
DOTP	Test variable used to determine if reflection is computed properly
DPH	Slope diffraction coefficient for the hard boundary condition
DPS	Slope diffraction coefficient for the soft boundary condition
DS	Diffraction coefficient for soft boundary condition
DV	Dot product of incident ray propagation vector and edge unit vector
EDPH	Phi component of edge-diffracted reflected E-field
EDPL	Component of diffracted field parallel to the edge

EDPR	Component of diffracted field perpendicular to the edge
EDTH	Theta component of edge-diffracted reflected E-field
EF	Source field pattern factor theta component
EG	Source field pattern factor phi component
EIPL	Component of incident field parallel to the edge or plane of incidence
EIPR	Component of incident field perpendicular to the edge or plane of incidence
EIX,EIY,EIZ	Source pattern factors for x,y, and z components of incident E-field
ERPP	Component of reflected E-field parallel to plane of incidence
ERPR	Component of reflected E-field perpendicular to plane of incidence
ERX,ERY,ERZ	X,Y,Z components of reflected field in RCS
EXPH	Complex phase and spreading factor
FN	Wedge angle number
LDRC	Set true if starting point information exists from previous pattern angle
LHIT	Set true if plate is hit
ME	Edge on plate MP where diffraction occurs
MP	Plate for which diffraction occurs
PD	Dot product of edge binormal and propagation direction
PH	Diffracted field phi unit vector perpendicular to edge
PHIR	Phi component of propagation direction of ray incident on plate MP

PHO	Incident field phi unit vector perpendicular to edge
PP	Negative dot product of edge binormal and incident ray unit vector
PS	Diffracted ray phi angle in edge-fixed coordinate system
PSO	Phi angle of incident ray direction in edge-fixed coordinate system in degrees
PSOR	Phi angle of incident ray direction in edge-fixed coordinate system
PSR	Phi component of diffracted ray propagation direction in edge-fixed coordinate system
QD	Dot product of plate normal and diffracted ray propagation direction
QI	Negative of dot product of plate normal and incident ray unit vector
RHI1	Radius of curvature perpendicular to edge of diffracted ray incident on reflection point
RHI2	Radius of curvature in edge plane of diffracted ray incident on reflection point
RHO1	Ray spreading radius in plane of cylinder curvature at reflection point
RHO2	Ray spreading radius in plane normal to plane of incidence at cylinder reflection point
S	Distance from source to reflection point
SBO	Sine of the diffraction angle
SMAG	Distance from diffraction point to reflection point
SNF	Distance between near-field observation point and reflection point on cylinder

SP	Distance from source to diffraction point (from subroutine DFRFPT)
TEMP	A temporary storage variable array
THIR	Theta component of propagation direction of ray incident on plate MP
TPP	The distance parameter
UB	X,Y components of unit vector tangent to cylinder at reflection point
UIPPX,UIPPY,UIPPZ	X,Y,Z components of incident field polarization unit vector parallel to plane of incidence
UIPRX,UIPRY,UIPRZ	X,Y,Z components of incident/reflected field polarization unit vector perpendicular to plane of incidence
UN	X,Y components of unit vector normal to cylinder at reflection point
URPPX,URPPY,URPPZ	X,Y,Z components of reflected field polarization unit vector parallel to plane of incidence
VI	Unit vector of propagation direction of ray incident on diffraction point (from subroutine DFRFPT)
VIC	X,Y,Z components of unit vector of ray direction between diffraction and reflection
VR	Elliptical angle defining reflection point on cylinder (2-D) in RCS x-y plane
VXS	3 X 3 matrix defining the source coordinate system axes
XD	X,Y,Z components of diffraction point
XDD	Diffraction point location
XDP	Modified diffraction point location
XR	X,Y,Z components of reflection point

XS	Source location in reference coordinate system
XSS	Source location

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
A	/GEOMEL/
B	/GEOMEL/
BD	/BDNFCL/
BTDC	/BNDDCL/
CPHS	/DIR/
CPS	/DIR/
CTC	/GEOMEL/
D	/DIR/
DDC	/BNDDCL/
DP	/THPHUV/
DPR	/PIS/
DT	/THPHUV/
DTDC	/BNDDCL/
FLDPT	/NEAR/
FN	F.P.
LDEBUG	/TEST/
LDRC	/CLDRC/
LNRFLD	/NEAR/
LUPRNT	/ADEBUG/
ME	F.P.

MP	F.P.
PHSR	/DIR/
PI	/PIS/
SPHS	/DIR/
SPS	/DIR/
THSR	/DIR/
TPI	/PIS/
V	/GEOPLA/
VN	/GEOPLA/
VP	/GEOPLA/
VXS	/SORINF/
XS	/SORINF/
ZC	/GEOMEL/

B. OUTPUT LOCATION

EDPH	F.P.
EDTH	F.P.
LDRC	/CLDRC/

6. CALLING ROUTINE:

GTDDRV

7. CALLED ROUTINES:

ASSIGN

BEXP

BTAN2

CYLINT

DFRFPT

DW

NANDB

NFD

PLAINT

RFDFPT

SMAGNF

SOURCE

STATIN

STATOT

TPNFLD

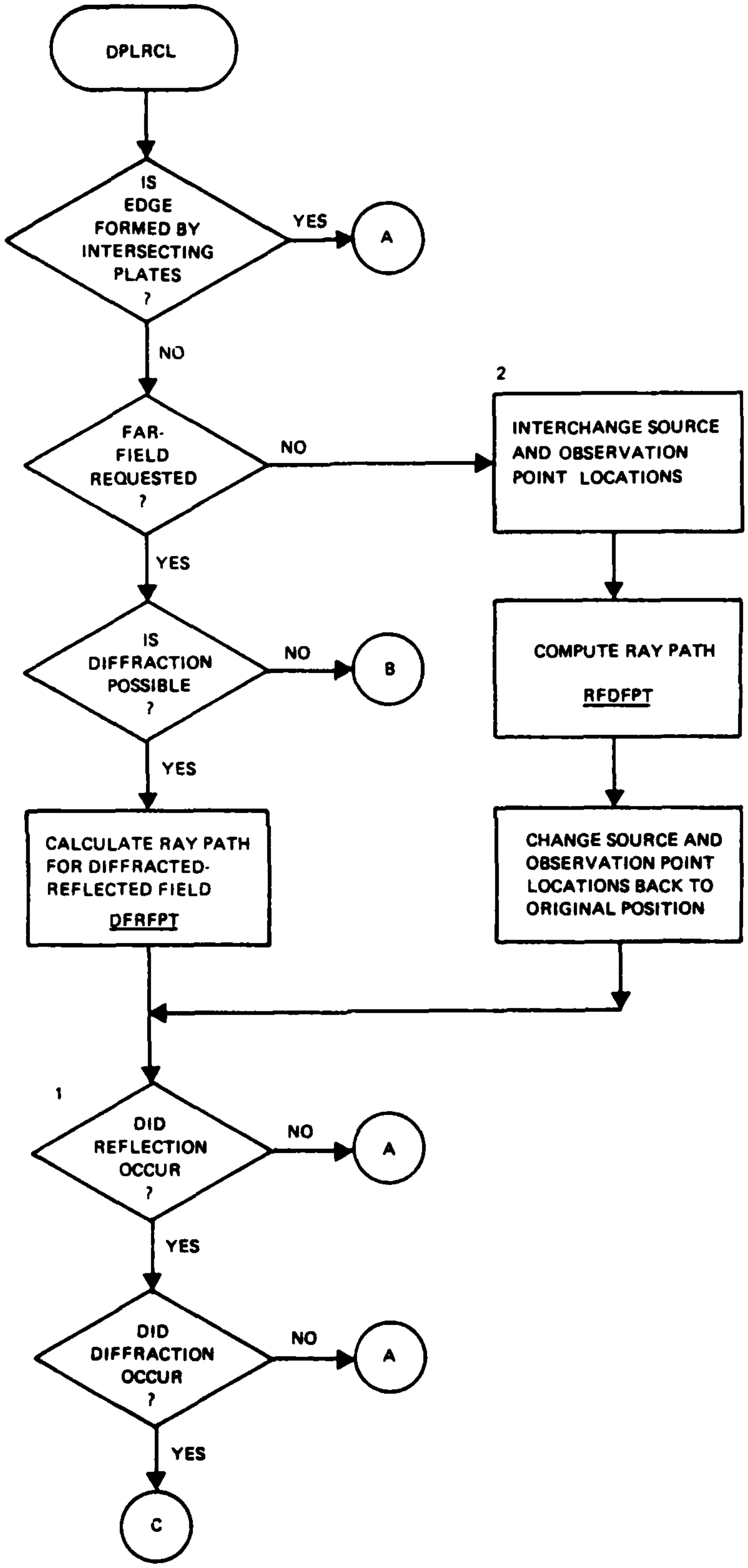
WLKBCK

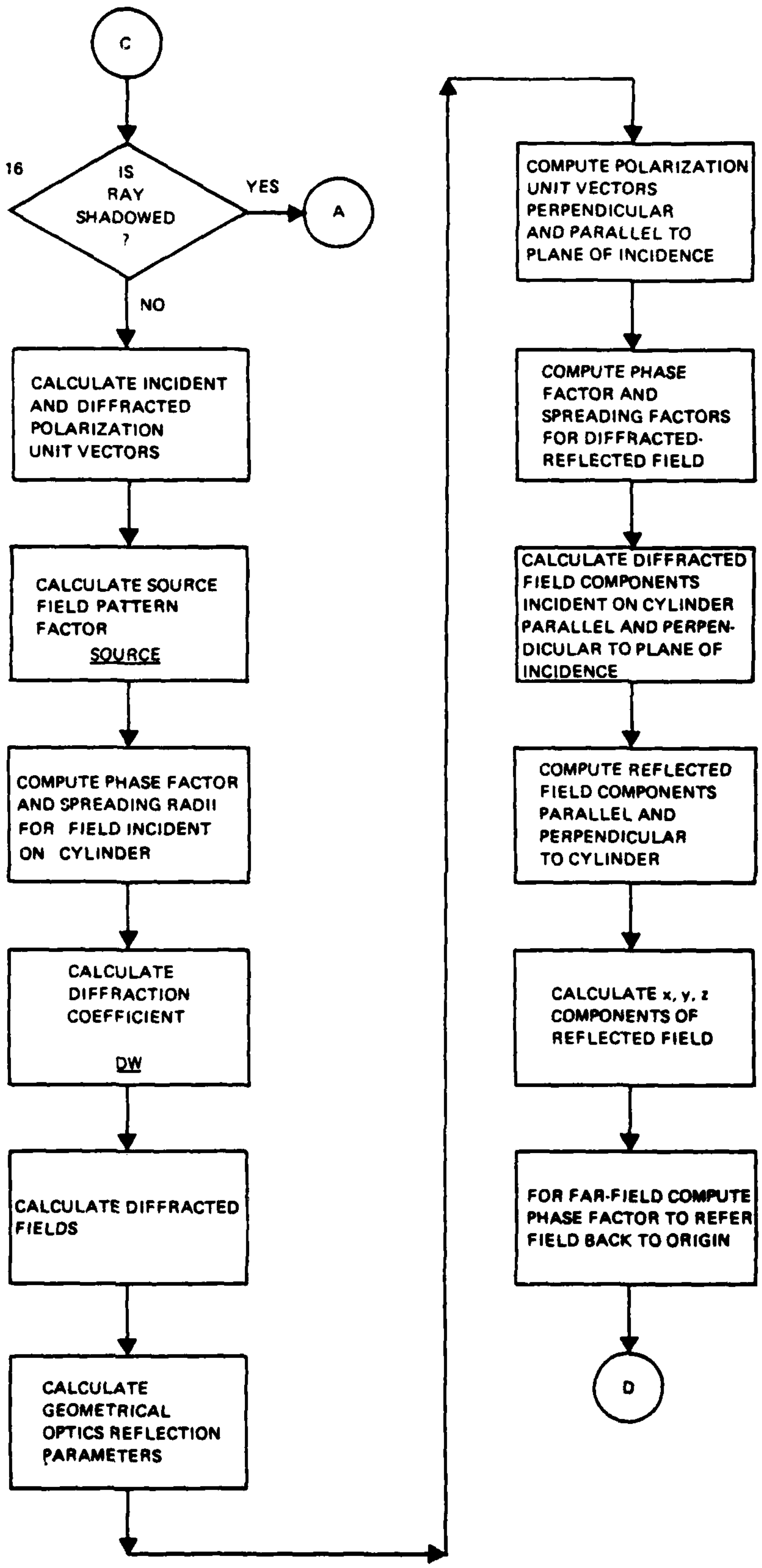
XYZFLD

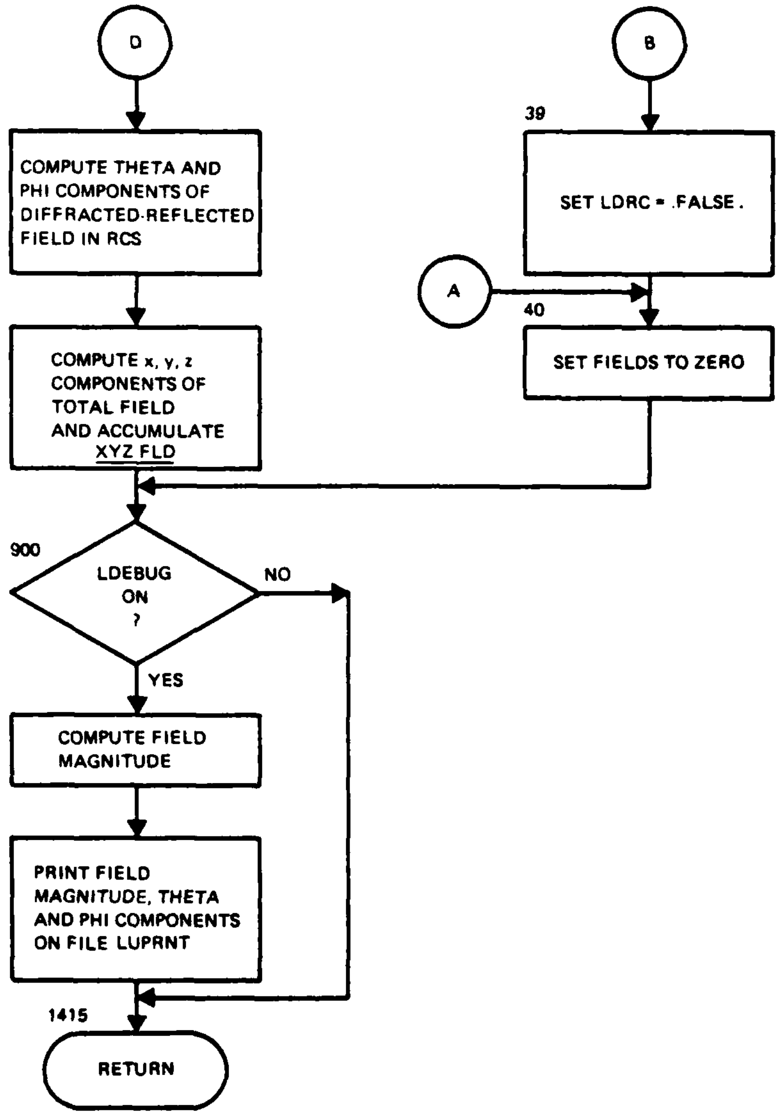
8. REFERENCES:

A. R. G. Kouyoumjian and P. H. Pathak, "A Uniform Geometrical Theory of Diffraction for an Edge in a Perfectly Conducting Surface," Proc. IEEE, Vol. 62, November 1974, pp. 1448-1461.

B. R. J. Marhefka, "Analysis of Aircraft Wing-Mounted Antenna Patterns," Report 2902-25, June 1976, The Ohio State University ElectroScience Laboratory, Department of Electrical Engineering; prepared under Grant No. NGL 36-008-138 for National Aeronautics and Space Administration.


DPLRCL
IS EDGE FORMED BY INTERSECTING PLATES ?
YES
A
NO
FAR-FIELD REQUESTED ?
NO
2
INTERCHANGE SOURCE AND OBSERVATION POINT LOCATIONS
COMPUTE RAY PATH
RFDFPT
CHANGE SOURCE AND OBSERVATION POINT LOCATIONS BACK TO ORIGINAL POSITION
YES
IS DIFFRACTION POSSIBLE ?
NO
B
YES
CALCULATE RAY PATH FOR DIFFRACTED-REFLECTED FIELD
DFRFPT
1
DID REFLECTION OCCUR ?
NO
A
YES
DID DIFFRACTION OCCUR ?
NO
A
YES
C


C
16
IS RAY SHADOWED ?
YES
A
NO
CALCULATE INCIDENT AND DIFFRACTED POLARIZATION UNIT VECTORS
CALCULATE SOURCE FIELD PATTERN FACTOR
SOURCE
COMPUTE PHASE FACTOR AND SPREADING RADII FOR FIELD INCIDENT ON CYLINDER
CALCULATE DIFFRACTION COEFFICIENT
DW
CALCULATE DIFFRACTED FIELDS
CALCULATE GEOMETRICAL OPTICS REFLECTION PARAMETERS
COMPUTE POLARIZATION UNIT VECTORS PERPENDICULAR AND PARALLEL TO PLANE OF INCIDENCE
COMPUTE PHASE FACTOR AND SPREADING FACTORS FOR DIFFRACTED-REFLECTED FIELD
CALCULATE DIFFRACTED FIELD COMPONENTS INCIDENT ON CYLINDER PARALLEL AND PERPENDICULAR TO PLANE OF INCIDENCE
COMPUTE REFLECTED FIELD COMPONENTS PARALLEL AND PERPENDICULAR TO CYLINDER
CALCULATE x, y, z COMPONENTS OF REFLECTED FIELD
FOR FAR-FIELD COMPUTE PHASE FACTOR TO REFER FIELD BACK TO ORIGIN
D


D
B
39
COMPUTE THETA AND PHI COMPONENTS OF DIFFRACTED-REFLECTED FIELD IN RCS
SET LDRC = .FALSE.
A
40
COMPUTE x, y, z COMPONENTS OF TOTAL FIELD AND ACCUMULATE XYZ FLD
SET FIELDS TO ZERO
900
LDEBUG ON ?
NO
YES
COMPUTE FIELD MAGNITUDE
PRINT FIELD MAGNITUDE, THETA AND PHI COMPONENTS ON FILE LUPRNT
1415
RETURN

1. NAME: DPLRPL (GTD)

2. PURPOSE: To compute the unobstructed electric field from a unit source diffracted by edge ME of plate MP and then reflected by plate MR into a given far-field direction or to a given near-field observation point.

3. METHOD: DPLRPL is the driver routine for the analysis of the edge diffracted then plate reflected field. Pertinent geometry is shown in figures 1 and 2. Computation details are given in references A, B, and C.

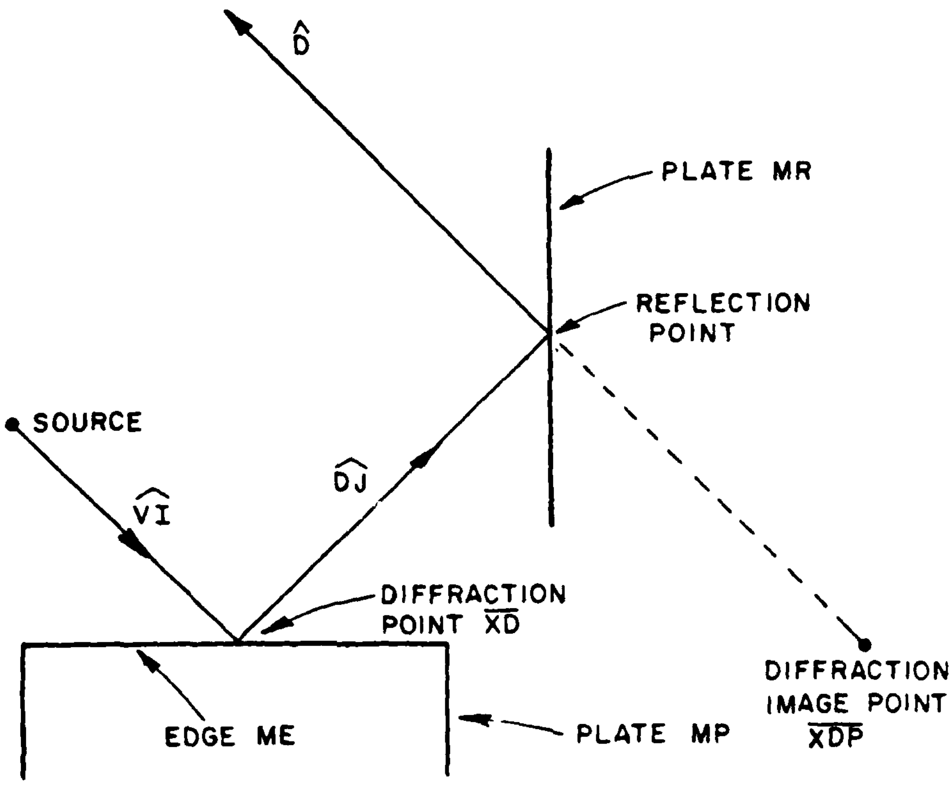


Figure 1. Illustration of Edge-Diffracted, Plate-Reflected Ray

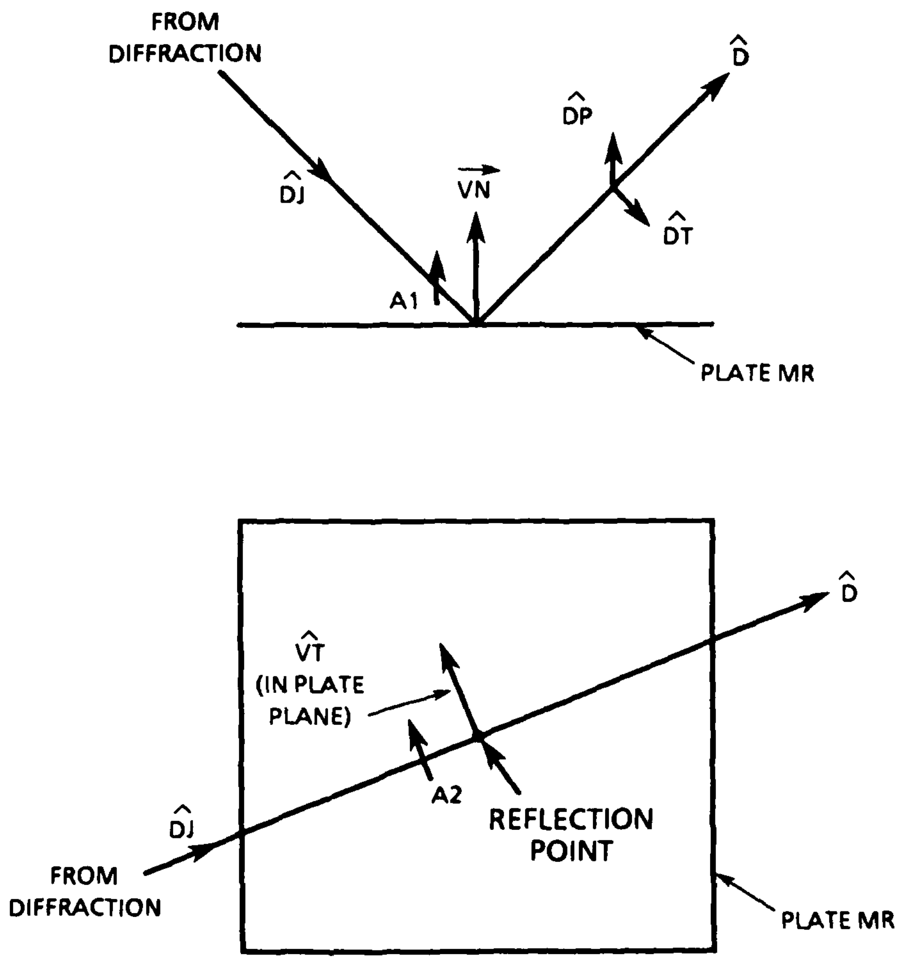


Figure 2. Geometry Used in Computing Plate Reflection

The fields are initialized to zero. To find the diffraction point on edge ME of plate MP, the direction the ray must travel for far-field calculations, or the point the ray must travel towards for near-field calculations, must be found first. The direction the far-field ray path would take between the diffraction and reflection plates is found by imaging the observation direction through plate MR. This is performed in subroutine REFBP. For a near-field case, the observation field point is imaged through plate MR. The diffracted ray would travel towards this image point. The image point is found in subroutine IMAGE. The diffraction point is found in subroutine DFPTWD. If diffraction did not occur, debug information (if requested) is printed and then control is returned to the calling routine. If diffraction does exist but the diffraction point is on the edge tangent outside of the corners, the point location is changed to that of the closest corner. A flag (LDIF=FALSE) is set to indicate that only corner diffraction is possible. Then the ray paths on plate MR are checked to make sure reflection occurs. This is followed by checking the complete ray path for obstructions. If reflection does not occur or if the ray path is shadowed, debug information (if requested) is printed on file LUPRNT and then control is returned to the calling routine. If reflection occurs and the ray path is clear of obstructions, the fields can be computed.

First the edge-diffracted field is computed. This includes the regular diffracted field; and, if slope diffraction was requested, the slope diffracted field. The source field pattern factor is found in subroutine SOURCE. The slope field pattern factor is found in subroutine SOURCP. The edge diffraction coefficient is found by calling subroutine DW. The diffracted field is found in components parallel and perpendicular to the diffracting edge and then in components normal and tangent to the reflecting plate. Then by multiplying by the correct polarization components and phase terms, the diffracted-reflected field is found in reference coordinate system (RCS) theta and phi components. Subroutine XYZFLD is called to compute the x, y, z components of the field and to accumulate it with fields from the other interactions.

If corner diffraction was requested, the far-field corner-diffracted fields are computed for each corner on edge ME. The same steps used in computing the edge diffracted fields are used for computing the corner diffracted fields.

If debug information was requested, the total field magnitude is computed. The magnitude, theta, and phi RCS components are printed on file LUPRNT. Control is returned to the calling routine.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
A1	Component of incident diffracted field normal to plate MR
A2	Component of incident diffracted field tangent to plate MR
A3	Determinant of transformation matrix
ADN	Dot product of vector from plate MP to the source and the plate unit normal
AFN	Wedge angle number
AN	Distance from plate MR plane to FLDPT and also distance from plate MR to the diffraction point on plate MP
BETN	Difference in diffracted and incident phi angles
BETP	Sum of diffracted and incident phi angles
BO	Diffracted field beta polarization unit vector (in edge-fixed coordinate system) in RCS components (for diffracting edge)
BOP	Incident field beta polarization unit vector (in edge-fixed coordinate system) in RCS components (for diffracting edge)
BRD	Array for the dot products of the unit edge vector and the unit vector between the source and each corner on edge ME (from subroutine DFPTWD) (not used in this subroutine)
C11	Dot product of reflected field polarization vector DT and plate coordinate system unit vector VN
C11A	Dot product of ray-fixed coordinate system vector BO and plate coordinate system vector VN

C12	Dot product of ray-fixed coordinate system vector DP and plate coordinate system unit vector VN
C12A	Dot product of ray-fixed coordinate system vector PH and plate coordinate system vector VN
C21	Dot product of ray-fixed coordinate system vector DT and plate coordinate system unit vector VT
C21A	Dot product of ray-fixed coordinate system vector BO and plate coordinate system vector VT
C22	Dot product of reflected field polarization unit vector DP and plate coordinate system unit vector VT
C22A	Dot product of ray-fixed coordinate system vector PH and plate coordinate system vector VT
CNP	Cosine of half wedge angle
CORN	Corner diffraction coefficient
CPH	Cosine of PSR
CPHJ	Cosine of PHJR
CPHO	Cosine of PSOR
CTH	Cosine of THR
CTHJ	Cosine of THJR
CTHP	Cosine of THPR
DEL	Parameter used in transition function
DH	Diffraction coefficient for hard boundary condition
DHIT	Distance from source to nearest hit point (from subroutine PLAINT or CYLINT)
DHT	Distance from source to hit point (returned from PLAINT and CYLINT)

DJ	X,Y, and Z components of ray propagation direction between diffraction and reflection
DMAG	Distance between reflection point and observation field point
DPH	Slope diffraction coefficient for hard boundary condition
DPS	Slope diffraction coefficient for soft boundary condition
DS	Diffraction coefficient for soft boundary condition
DV	Dot product of edge vector and diffracted ray propagation direction unit vector DJ
ECBI	Diffraction coefficient (from subroutine DICOEF) for incident diffracted field, modified for corner diffraction
ECBR	Edge diffraction coefficient (from subroutine DICOEF) for reflected diffracted field, modified for corner diffraction
ECPH	Phi component of corner diffracted, reflected E-field
ECTH	Theta component of corner diffracted, reflected E-field
EDPH	Phi component of edge diffracted, reflected E-field
EDPL	Component of diffracted field parallel to the edge
EDPR	Component of diffracted field perpendicular to the edge
EDTH	Theta component of edge diffracted, reflected E-field
EF	Theta component of pattern factor of field incident on edge - also theta component of reflected field in RCS

EG	Phi component of pattern factor of field incident on edge - also phi component of reflected field in RCS
EIPL	Component of incident field parallel to the edge
EIPLP	Pattern factor for component of source (incident) slope field parallel to the edge (ray incident on diffracting edge)
EIPR	Component of incident field perpendicular to the edge
EIPRP	Pattern factor for component of source (incident) slope field perpendicular to the edge (ray incident on diffracting edge)
EIX,EIY,EIZ	Source pattern factors for x,y, and z components of incident field on edge
EXPH	Complex phase term
FLDPTI	Location of field point imaged through plate MR
FN	Wedge angle number
FNN	Wedge angle indicator
FNP	Angle exterior to wedge in degrees
GAM	Dot product of the propagation direction and the vector from the RCS origin to the diffraction point image location
ISN	Sign change variable
LDIF	Logical variable set false if diffraction point is on edge tangent outside corners (Diffraction point moved to closest corner)
LDIFFR	Logical variable set true if edge diffraction occurred (from subroutine DFPTWD)
LHIT	Set true if ray hits a plate or cylinder (from PLAINT or CYLINT)
MC	Corner at end of edge ME

ME	Edge on plate MP where diffraction occurs
MP	Plate for which diffraction occurs
MR	Plate where reflection occurs
N	DO loop variable
PD	Dot product of edge binormal and propagation direction
PH	Diffracted field phi polarization unit vector (in edge-fixed coordinate system) in RCS components (for diffracting edge)
PHIR	Phi component of incident ray direction in RCS
PHJR	Phi component of ray propagation direction between diffraction and reflection in RCS
PHO	Incident field phi polarization unit vector (in edge-fixed coordinate system) in RCS components (for diffracting edge)
PHSR	Phi component of propagation direction after reflection in RCS
PP	Negative dot product of edge binormal and incident ray unit normal
PS	PSR*DPR
PSD	Diffracted ray phi angle in edge-fixed coordinate system
PSO	PSOR*DPR
PSOD	Incident ray phi angle in edge-fixed coordinate system
PSOR	Phi component of incident ray direction in edge-fixed coordinate system
PSR	Phi component of diffracted ray direction in edge-fixed coordinate system
QD	Dot product of plate normal and propagation direction

QI	Negative of dot product of plate normal and incident ray propagation direction
RM	Magnitude of vector from corner MC to source
RX,RY,RZ	X,Y, and Z components of vector from corner MC to source
SBO	Sine of BO, the angle the diffracted ray makes with the edge
SNF	Distance between diffraction point and near-field observation point imaged through plate MR
SNP	Sine of half wedge angle
SP	Distance from source to diffraction point (from subroutine DFPTWD)
SPH	Sine of PSR
SPHJ	Sine of PHJR
SPHO	Sine of PSOR
SPP	Distance from source to modified diffraction point
STHJ	Sine of THJR
STHR	Sine of THR
TERM	Coefficient of corner diffracted fields
THIR	Theta component of incident ray direction in RCS
THJR	Theta component of ray propagation direction between diffraction and reflection in RCS
THPR	Angle diffracted ray makes with edge
THR	Angle between edge unit vector and ray from source to corner MC

TPP	Distance parameter used in calculating diffraction coefficients
VCM	Array which contains the distance between the source and each corner on edge ME (from subroutine DFPTWD) (not used in this subroutine)
VECT	Vector used to move diffraction point off edge for shadowing tests
VI	Unit vector of ray incident on edge from source (from subroutine DFPTWD)
VIP	Unit vector from source to modified diffraction point
VMG	Distance along the edge from first corner of edge ME to diffraction point
VT	X,Y, and Z components of unit vector on plate MR normal to plane of incidence (tangent to plate)
VXS	3 X 3 matrix defining the source coordinate system axes
XD	Diffraction point (calculated in subroutine DFPTWD)
XD1	Diffraction point location
XDI	Diffraction point image through plate MR
XDP	Modified diffraction point used for shadowing tests--also location of diffraction point image in plate MR
XDPP	Diffraction point, converted to reflection hit point
XS	Source location in RCS
XS1	Source location
XSS	Source location
ZP	Dot product of propagation unit vector and vector from diffraction point to corner MC

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
D	/DIR/
DP	/THPHUV/
DPR	/PIS/
DT	/THPHUV/
FLDPT	/NEAR/
FNN	F.P.
LCORNR	/LOGDIF/
LDEBUG	/TEST/
LNRFLD	/NEAR/
LSLOPE	/LOGDIF/
LSURF	/SURFAC/
LUPRNT	/ADEBUG/
ME	F.P.
MEP	/GEOPLA/
MP	F.P.
MR	F.P.
PHSR	/DIR/
PI	/PIS/
THSR	/DIR/
TPI	/PIS/
V	/GEOPLA/
VMAG	/EDMAG/
VN	/GEOPLA/

VP	/GEOPLA/
VXS	/SORINF/
X	/GEOPLA/
XS	/SORINF/

B. OUTPUT — LOCATION

ECPH	F.P.
ECTH	F.P.
EDPH	F.P.
EDTH	F.P.

6. CALLING ROUTINE:

GTDDRV

7. CALLED ROUTINES:

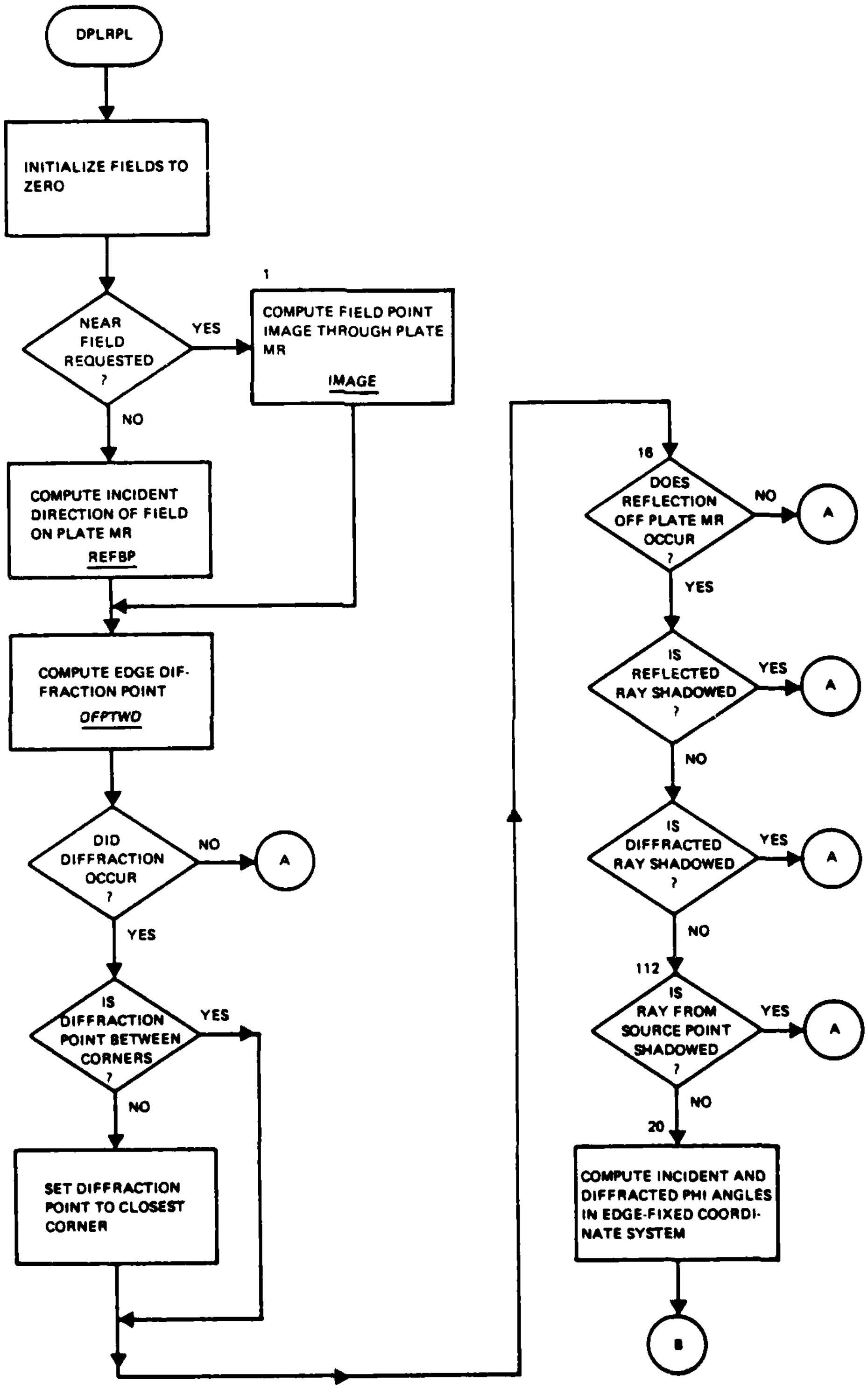
ASSIGN	PLAINT
BEXP	REFBP
BTAN2	SMAGNF
CYLINT	SOURCE
DFPTWD	SOURCP
DICOEF	STATIN
DW	STATOT
FFCT	TPNFLD
IMAGE	WLKBCK
NFD	XYZFLD

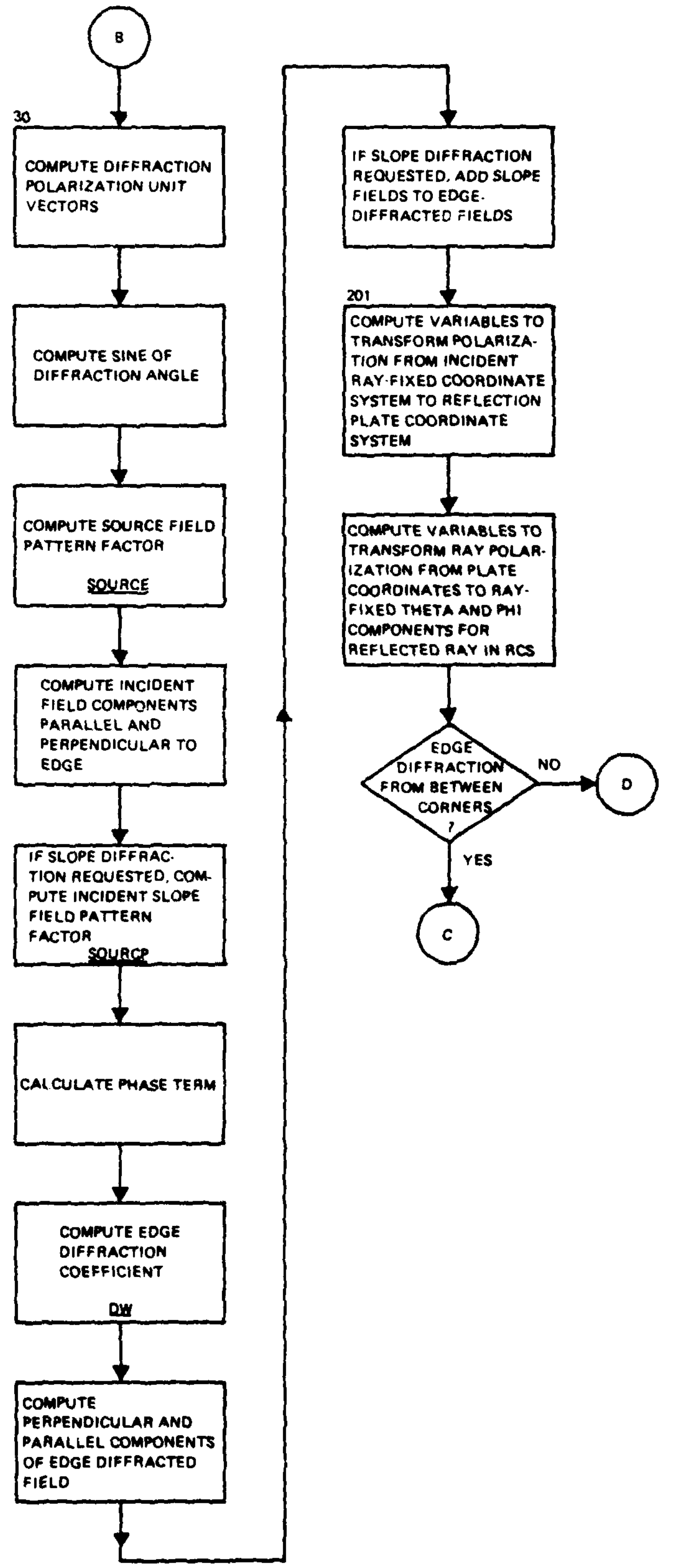
8. REFERENCES:

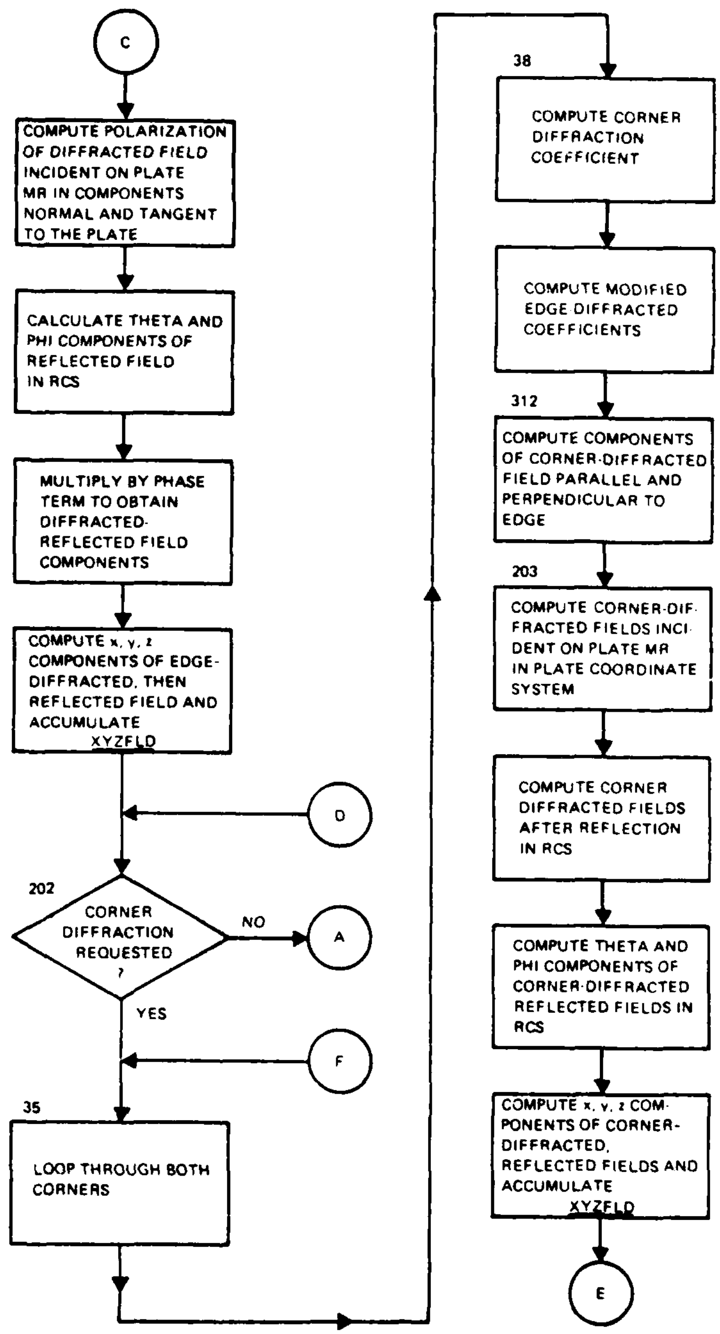
A. R. G. Kouyoumjian and P. H. Pathak, "A Uniform Geometrical Theory of Diffraction for an Edge in a Perfectly Conducting Surface," Proc. IEEE, Vol. 62, November 1974, pp. 1448-1461.

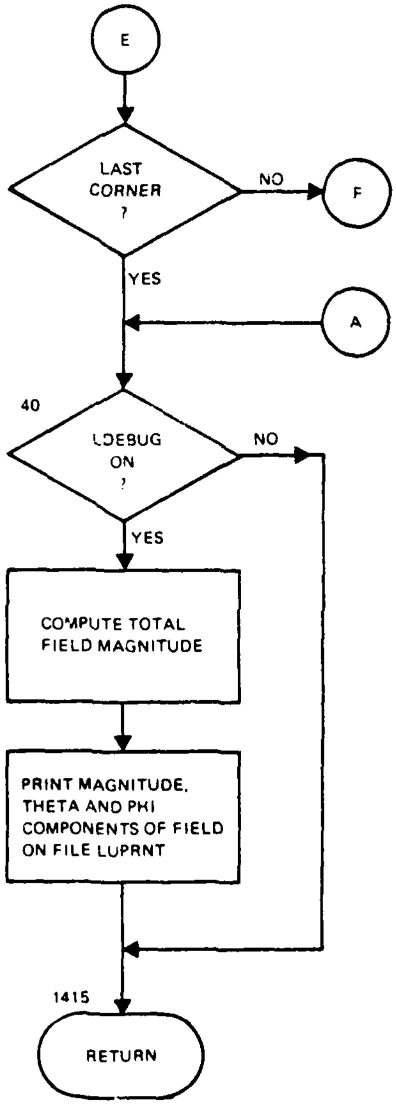
B. W. D. Burnside and P. H. Pathak, "A Corner Diffraction Coefficient," to appear.

C. Y. M. Hwang and R. G. Kouyoumjian, "A Dyadic Diffraction Coefficient for an Electromagnetic Wave Which Is Rapidly Varying at an Edge," USNC-URSI 1974 Annual Meeting, Boulder, CO., Oct. 1974.


DPLRPL
INITIALIZE FIELDS TO ZERO
NEAR FIELD REQUESTED ?
YES
1
COMPUTE FIELD POINT IMAGE THROUGH PLATE MR
IMAGE
NO
COMPUTE INCIDENT DIRECTION OF FIELD ON PLATE MR
REFBP
COMPUTE EDGE DIFFRACTION POINT
DFPTWD
DID DIFFRACTION OCCUR ?
NO
A
YES
IS DIFFRACTION POINT BETWEEN CORNERS ?
YES
NO
SET DIFFRACTION POINT TO CLOSEST CORNER
16
DOES REFLECTION OFF PLATE MR OCCUR ?
NO
A
YES
IS REFLECTED RAY SHADOWED ?
YES
A
NO
IS DIFFRACTED RAY SHADOWED ?
YES
A
NO
112
IS RAY FROM SOURCE POINT SHADOWED ?
YES
A
NO
20
COMPUTE INCIDENT AND DIFFRACTED PHI ANGLES IN EDGE-FIXED COORDINATE SYSTEM
B


B
30
COMPUTE DIFFRACTION POLARIZATION UNIT VECTORS
COMPUTE SINE OF DIFFRACTION ANGLE
COMPUTE SOURCE FIELD PATTERN FACTOR
SOURCE
COMPUTE INCIDENT FIELD COMPONENTS PARALLEL AND PERPENDICULAR TO EDGE
IF SLOPE DIFFRACTION REQUESTED, COMPUTE INCIDENT SLOPE FIELD PATTERN FACTOR
SOURCP
CALCULATE PHASE TERM
COMPUTE EDGE DIFFRACTION COEFFICIENT
DW
COMPUTE PERPENDICULAR AND PARALLEL COMPONENTS OF EDGE DIFFRACTED FIELD
IF SLOPE DIFFRACTION REQUESTED, ADD SLOPE FIELDS TO EDGE-DIFFRACTED FIELDS
201
COMPUTE VARIABLES TO TRANSFORM POLARIZATION FROM INCIDENT RAY-FIXED COORDINATE SYSTEM TO REFLECTION PLATE COORDINATE SYSTEM
COMPUTE VARIABLES TO TRANSFORM RAY POLARIZATION FROM PLATE COORDINATES TO RAY-FIXED THETA AND PHI COMPONENTS FOR REFLECTED RAY IN RCS
EDGE DIFFRACTION FROM BETWEEN CORNERS ?
NO
D
YES
C


C
COMPUTE POLARIZATION OF DIFFRACTED FIELD INCIDENT ON PLATE MR IN COMPONENTS NORMAL AND TANGENT TO THE PLATE
CALCULATE THETA AND PHI COMPONENTS OF REFLECTED FIELD IN RCS
MULTIPLY BY PHASE TERM TO OBTAIN DIFFRACTED-REFLECTED FIELD COMPONENTS
COMPUTE x, y, z COMPONENTS OF EDGE-DIFFRACTED, THEN REFLECTED FIELD AND ACCUMULATE XYZFLD
D
202
CORNER DIFFRACTION REQUESTED ?
NO
A
YES
F
35
LOOP THROUGH BOTH CORNERS
38
COMPUTE CORNER DIFFRACTION COEFFICIENT
COMPUTE MODIFIED EDGE DIFFRACTED COEFFICIENTS
312
COMPUTE COMPONENTS OF CORNER-DIFFRACTED FIELD PARALLEL AND PERPENDICULAR TO EDGE
203
COMPUTE CORNER-DIFFRACTED FIELDS INCIDENT ON PLATE MR IN PLATE COORDINATE SYSTEM
COMPUTE CORNER DIFFRACTED FIELDS AFTER REFLECTION IN RCS
COMPUTE THETA AND PHI COMPONENTS OF CORNER-DIFFRACTED REFLECTED FIELDS IN RCS
COMPUTE x, y, z COMPONENTS OF CORNER-DIFFRACTED, REFLECTED FIELDS AND ACCUMULATE XYZFLD
E


E
LAST
CORNER
?
NO
F
YES
A
40
LDEBUG
ON
?
NO
YES
COMPUTE TOTAL
FIELD MAGNITUDE
PRINT MAGNITUDE,
THETA AND PHI
COMPONENTS OF FIELD
ON FILE LUPRNT
1415
RETURN

1. NAME: DPTNFW (GTD)

2. PURPOSE: To compute the diffraction point for a ray which is diffracted by a given edge and observed at a specified near-field point in the vicinity of the plate.

3. METHOD: The diffraction point is found by using geometrical relationships involving similar triangles defined by perpendiculars from the source and observation points to the edge line. The quantities and geometries used are shown in figure 1.

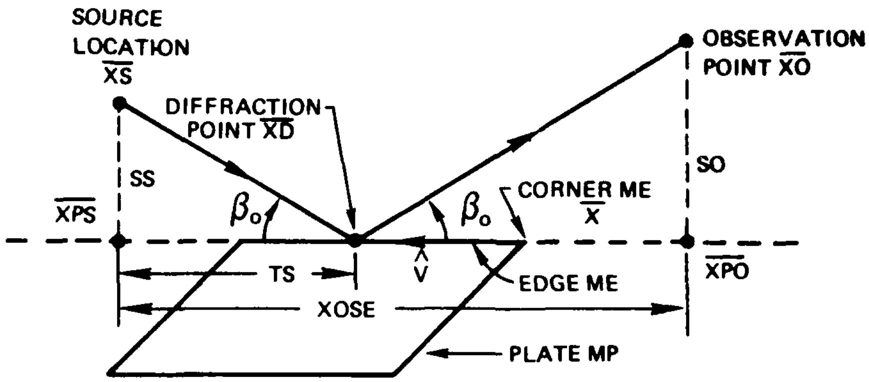


Figure 1. Geometry for Finding the Diffraction Point with the Observation Point in the Near Field of the Plate

The perpendicular from the source to the edge line is given by:

$$\overline{XPS} = XSCE\ \hat{V} + \overline{X}$$

where

$$XSCE = (\overline{XS} - \overline{X}) \cdot \hat{V}$$

and $\overline{X}$ is the vector to corner ME.

The perpendicular from the observation point to the edge line is given by:

$$\overline{XPO} = XOCE\ \hat{V} + \overline{X}$$

where

$$XOCE = (\overline{XO} - \overline{X}) \cdot \hat{V}$$

and $\overline{X}$ is the vector to corner ME.

The triangles are similar due to Snell's law: the angle of incidence equals the angle of diffraction. By enforcing Snell's law and calculating the distances SS, SO and XOSE given by

$$SS = |\overline{XS} - \overline{XPS}|$$

$$SO = |\overline{XO} - \overline{XPO}|$$

$$XOSE = (\overline{XO} - \overline{XS}) \cdot \hat{V}\ ,$$

the distance TS can be found by the following approach:

$$\tan \beta_o = \tan \beta_o$$

$$\frac{SS}{TS} = \frac{SO}{XOSE-TS}$$

$$(SS)(XOSE) - (SS)(TS) = (SO)(TS)$$

$$SS(XOSE) = TS\ (SO+SS)$$

$$TS = \frac{(SS)(XOSE)}{SO + SS}.$$

And by knowing also that

$$TS = |\overline{XD} - \overline{XPS}|\ ,$$

the diffraction point can be found by

$$\overline{XD} = \overline{XPS} + TS\ \hat{V}\ .$$

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
ME	Edge on plate MP where diffraction occurs
MP	Plate where diffraction occurs
SO	Distance from observation point to point XPO
SS	Distance from source to point XPS
TS	Distance from XPS to XD along edge line
V	Tangent vector to edge where diffraction occurs
X	X,Y,Z components of corner locations on edge ME of plate MP
XD	X,Y,Z components of diffraction point location in RCS
XO	X,Y,Z components of observation point in RCS
XOCE	Dot product of ray from corner ME to observation point and edge unit vector
XOSE	Dot product of ray from source to observation point and edge unit vector
XPO	Point on line through edge ME closest to observation point
XPS	Point on line through edge ME closest to source
XS	X,Y,Z components of source location in RCS
XSCE	Dot product of ray from corner ME to source and edge unit vector

5. I/O VARIABLES:

A. INPUT	LOCATION
ME	F.P.
MP	F.P.
V	/GEOPLA/
X	/GEOPLA/
XO	F.P.
XS	F.P.

B. OUTPUT	LOCATION
XD	F.P.

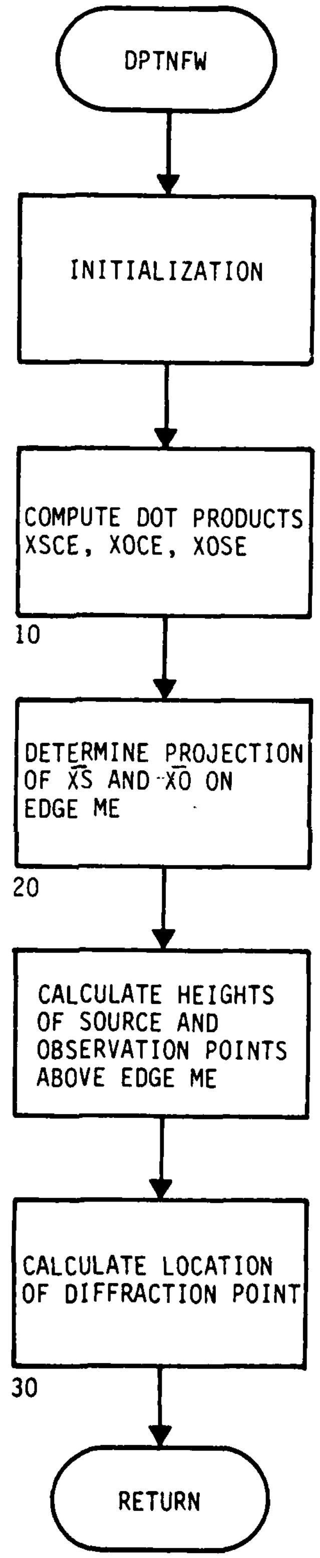
6. CALLING ROUTINE:

GEOMPC

7. CALLED ROUTINE:

None

DPTNFW (GTD)


DPTNFW
INITIALIZATION
COMPUTE DOT PRODUCTS XSCE, XOCE, XOSE
10
DETERMINE PROJECTION OF $\overline{XS}$ AND $\overline{XO}$ ON EDGE ME
20
CALCULATE HEIGHTS OF SOURCE AND OBSERVATION POINTS ABOVE EDGE ME
CALCULATE LOCATION OF DIFFRACTION POINT
30
RETURN

1. NAME: DQG32 (GTD)

2. PURPOSE: To numerically integrate a given function over a specified range.

3. METHOD: This subroutine uses a 32 point Gaussian quadrature formula to compute the integral of a function. The form of the integral is given as (see reference A):

$$Y = \int_{XL}^{XU} FCT(x)dx.$$

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
FCT	Function defining the integrand
XL	Lower limit of integration
XU	Upper limit of integration
Y	Result of integral

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
FCT	F.P.
XL	F.P.
XU	F.P.

B. OUTPUT

OUTPUT	LOCATION
Y	F.P.

6. CALLING ROUTINES:

FKARG

RPLSCL

SCLRPL

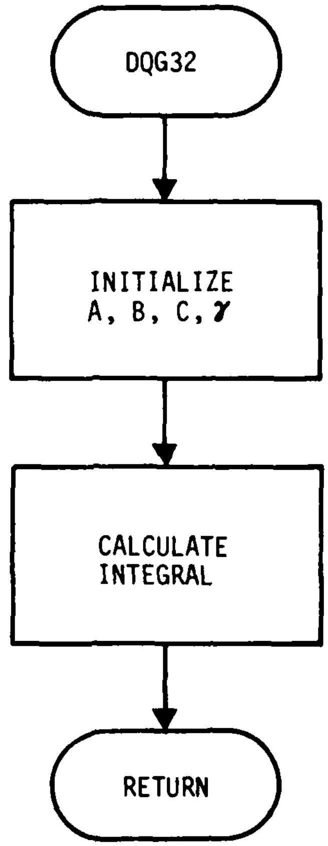
SCTCYL

7. CALLED ROUTINES:

   FCT

8. REFERENCE:

A. M. Abramowitz and I. A. Stegun, Handbook of Mathematical Functions, 7th Edition, Dover Publications, Inc., New York, 1970 pp. 887-888.


DQG32
INITIALIZE
A, B, C, γ
CALCULATE
INTEGRAL
RETURN

1. NAME: DW (GTD)

2. PURPOSE: To determine wedge and slope diffraction coefficients for the soft and hard boundary conditions.

3. METHOD: This subroutine directs the calculations of the edge diffraction and slope diffraction coefficients for the hard and soft boundary conditions using the Uniform Geometrical Theory of Diffraction (see references A and B). Subroutine DICOEF is called to calculate the edge diffraction coefficient. Subroutine DPI is called to calculate the slope diffraction coefficient. Subroutine DW sums the results appropriately.

The edge diffraction coefficient has the form:

$$D_{\substack{h\\s}} = DI(R,\phi-\phi',\sin\beta_o,n) \pm DI(R,\phi+\phi',\sin\beta_o,n),$$

where D_h is for the hard case and D_s is for the soft case and n is the wedge angle number (FN). The other variables are defined as shown in figure 1. See subroutine DICOEF for the actual calculations.

The slope diffraction coefficient has the form:

$$\frac{\partial D_{\substack{h\\s}}}{\partial\phi'} = DPI(R,\phi-\phi', \sin\beta_o,n) \mp DPI(R,\phi+\phi',\sin\beta_o,n)$$

See subroutine DPI for the actual calculations.

In both cases the $\phi-\phi'$ part refers to the incident part of the diffraction coefficient and $\phi+\phi'$ refers to the reflection part. For grazing incidence where $\phi'=0$, the diffraction coefficients have the form:

$$D_h = DI(R,\phi,\sin\beta_o,n)$$

$$D_s = 0$$

$$\frac{\partial D_s}{\partial\phi'} = DPI(R,\phi,\sin\beta_o,n)$$

$$\frac{\partial D_h}{\partial\phi'} = 0.$$

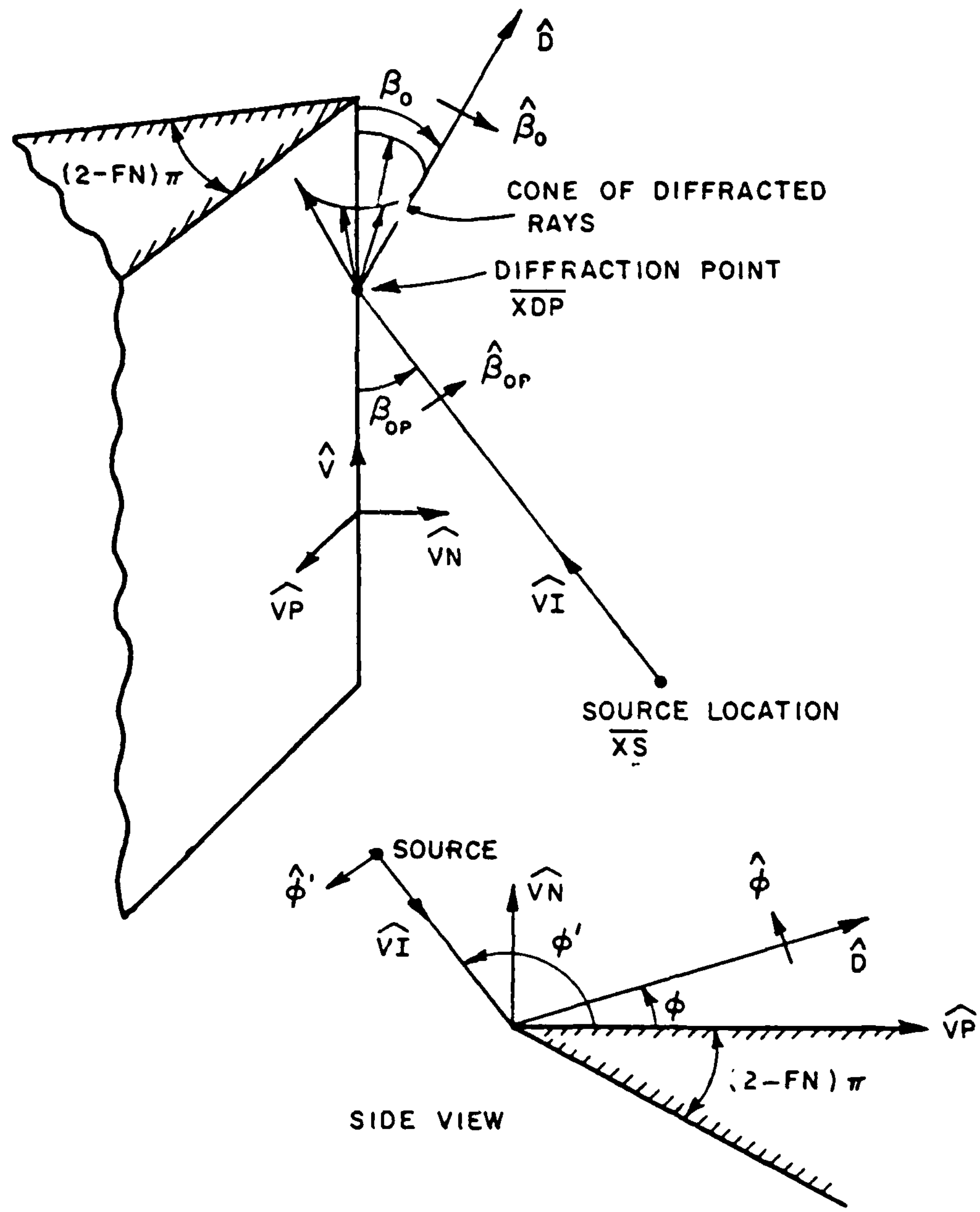


Figure 1. Edge Diffraction Geometry

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
BETN	Difference between diffraction and incidence angle
BETP	Difference between diffraction and image of incidence angle
DH	Edge diffraction coefficient for hard boundary condition
DIN	Incident part of edge diffraction coefficient
DIP	Reflection part of edge diffraction coefficient
DPH	Slope diffraction coefficient for hard boundary condition
DPN	Incident part of slope diffraction coefficient
DPP	Reflection part of slope diffraction coefficient
DPS	Slope diffraction coefficient for soft boundary condition
DS	Edge diffraction coefficient for soft boundary condition
FN	Wedge angle number
LSURF	A logical variable that is set true if the source is mounted on the surface of the wedge (grazing incidence)
PH	Diffraction ray phi angle in edge-fixed coordinate system (in degrees)
PHP	Incident ray phi angle in edge-fixed coordinate system (in degrees)

DW (GTD)

R	Distance parameter which is a function of source type, incidence angle, and source-to-diffraction point separation distance
SBO	Sine of β_O, the angle the rays make with the edge

5. I/O VARIABLES:

A. INPUT	LOCATION
FN	F.P.
LSURF	F.P.
PH	F.P.
PHP	F.P.
R	F.P.
SBO	F.P.

B. OUTPUT	LOCATION
DH	F.P.
DPH	F.P.
DPS	F.P.
DS	F.P.

6. CALLING ROUTINES:

DIFPLT

DPLRCL

DPLRPL

RCLDPL

RPLDPL

7. CALLED ROUTINES:

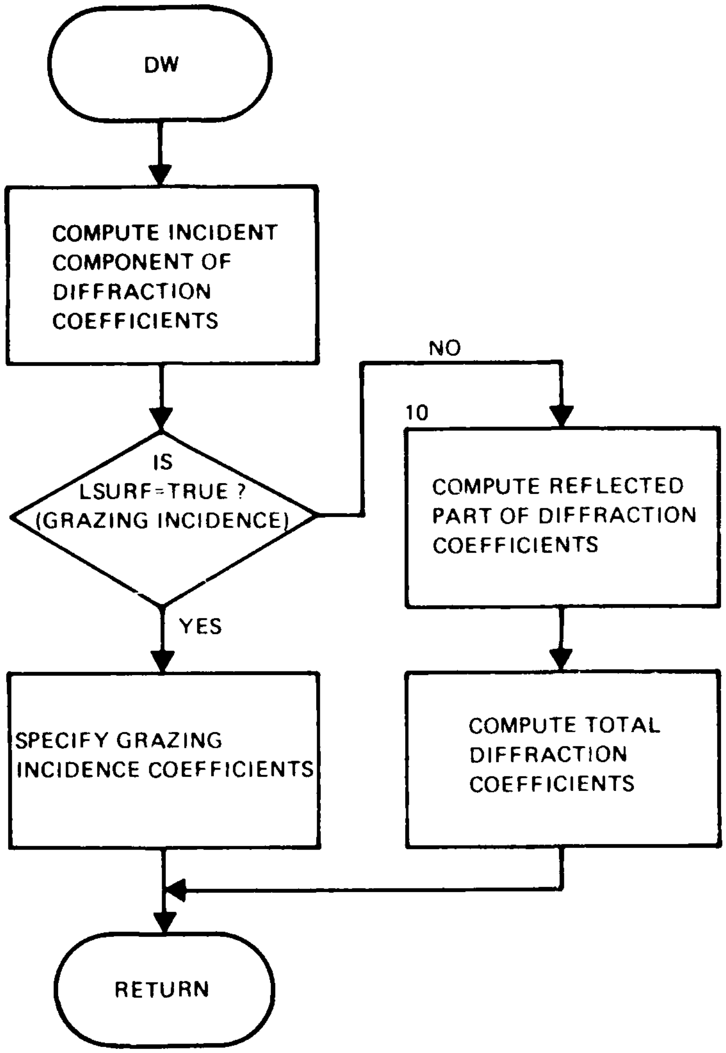
DICOEF

DPI

8. REFERENCES:

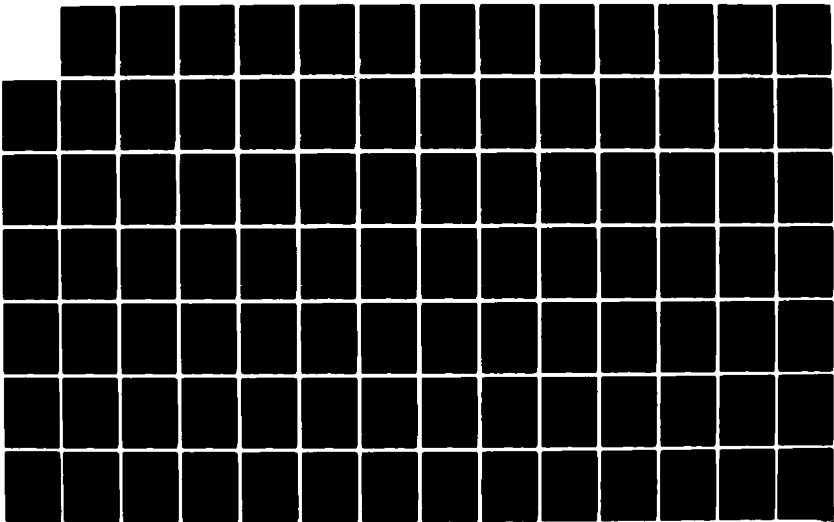
A. R.G. Kouyoumjian and P.H. Pathak, "A Uniform Geometrical Theory cf Diffraction for an Edge in a Perfectly Conducting Surface," Proc. IEEE, Vol. 62, November 1974, pp. 1448-1461.

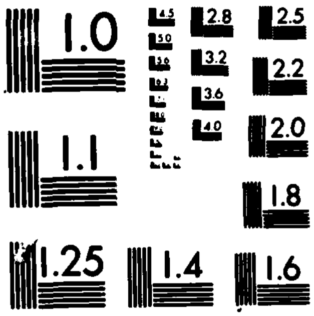
B. R.G. Kouyoumjian, "The Geometrical Theory of Diffraction and Its Applications," Numerical and Asymptotic Techniques in Electromagnetics, edited by R. Mittra, Spring-Verlag, New York, 1975.


DW
COMPUTE INCIDENT COMPONENT OF DIFFRACTION COEFFICIENTS
IS LSURF=TRUE ? (GRAZING INCIDENCE)
NO
10
COMPUTE REFLECTED PART OF DIFFRACTION COEFFICIENTS
YES
SPECIFY GRAZING INCIDENCE COEFFICIENTS
COMPUTE TOTAL DIFFRACTION COEFFICIENTS
RETURN

AD-A137 463 GENERAL ELECTROMAGNETIC MODEL FOR THE ANALYSIS OF COMPLEX SYSTEMS (GEMACS..(U) BDM CORP ALBUQUERQUE NM D L KADLEC ET AL. SEP 83 BDM/A-83-020-TR 4/5

UNCLASSIFIED RADC-TR-83-217-VOL-3 F30602-81-C-0084 F/G 20/14 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

1. NAME: DZCOEF (GTD)

2. PURPOSE: To compute the diffraction coefficient for an edge formed by two curved surfaces.

3. METHOD: This subroutine computes the diffraction coefficient for a curved edge based on the Uniform Geometrical Theory of Diffraction (see reference A). The diffraction coefficient is given by:

$$\underbrace{D_{\substack{s\\h}}(\phi,\phi',\beta_o)}_{DS,\ DH} = \underbrace{\frac{e^{-j\pi/4}}{2n\sqrt{2\pi k}\ \sin\beta_o}}_{F1} \left[\underbrace{\left[\frac{2\ \sin(\pi/n)F[kL^i a^-(\phi-\phi')]}{\cos(\pi/n)-\cos[(\phi-\phi')/n]}\right]}_{F2}\right.$$

$k\equiv 2\pi$ since units are normalized to wavelengths; thus $2\pi k \sim 2\pi$

$$\left.\pm\left\{\underbrace{\cot\left(\frac{\pi+(\phi+\phi')}{2n}\right)\ F[kL^{rn}a^+(\phi+\phi')]}_{F3} + \underbrace{\cot\left(\frac{\pi-(\phi+\phi')}{2n}\right)\ F[kL^{ro}a^-(\phi+\phi')]}_{F4}\right\}\right]$$

where

$a^-(\beta) = 2\cos^2\beta/2$ (denoted by AP)

$a^+(\beta) = 2\cos^2(2\pi n-\beta)/2$ (denoted by A)

and n is the wedge number (FN) and L^i, L^{rn}, L^{ro} are the distance parameters for the incident part, reflection from the n-surface and o-surface respectively. The distance parameters are functions of source type, incident angle, and source-to-diffraction point separation distance and are such to make the field continuous at shadow and reflection boundaries.

When the diffraction angle is close to one of the shadow boundaries, the following approximation is used

$$\cot\left(\frac{\pi\pm\beta}{2n}\right)\ F[kLa^{\pm}(\beta)] = \pm\sqrt{n\ 2\pi kL}\ e^{j\pi/4}\ e^{jk|L|a^{\pm}},$$

where the plus or minus sign is chosen depending on which side of the shadow boundary the diffraction angle is on. The variable $a^{+}(\beta)$ is denoted as A, as previously defined. The variable $a^{-}(\beta)$ is denoted as AP as previously defined.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
A	Angle function for incident and o-surface transition functions
AP	Angle function for n-surface transition function
CSP	COS(PMR/2.)
DH	Diffraction coefficient for hard boundary condition
DS	Diffraction coefficient for soft boundary condition
F1	Constant factor
F2	Incident part of diffraction coefficient
F3	N-surface part of diffraction coefficient
F4	O-surface part of diffraction coefficient
FLI	Distance parameter for the incident component
FLRN	Distance parameter for the reflection from the N-surface
FLRO	Distance parameter for the reflection from the O-surface
FN	Wedge angle number
PHPR	Incident ray angle in radians
PHR	Diffracted ray angle in radians
PI	π
PMR	Difference between diffraction angle and the incidence angle

PPR	Difference between diffraction angle and the image of the incidence angle
SBO	Sine of BO
TAN1	N-surface angular dependence of diffraction coefficient
TAN2	O-surface angular dependence of diffraction coefficient
TPI	2π

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
FLI	F.P.
FLRN	F.P.
FLRO	F.P.
FN	F.P.
PHR	F.P.
PHPR	F.P.
PI	/PIS/
SBO	F.P.
TPI	/PIS/

B. OUTPUT

OUTPUT	LOCATION
DH	F.P.
DS	F.P.

6. CALLING ROUTINE: ENDIF

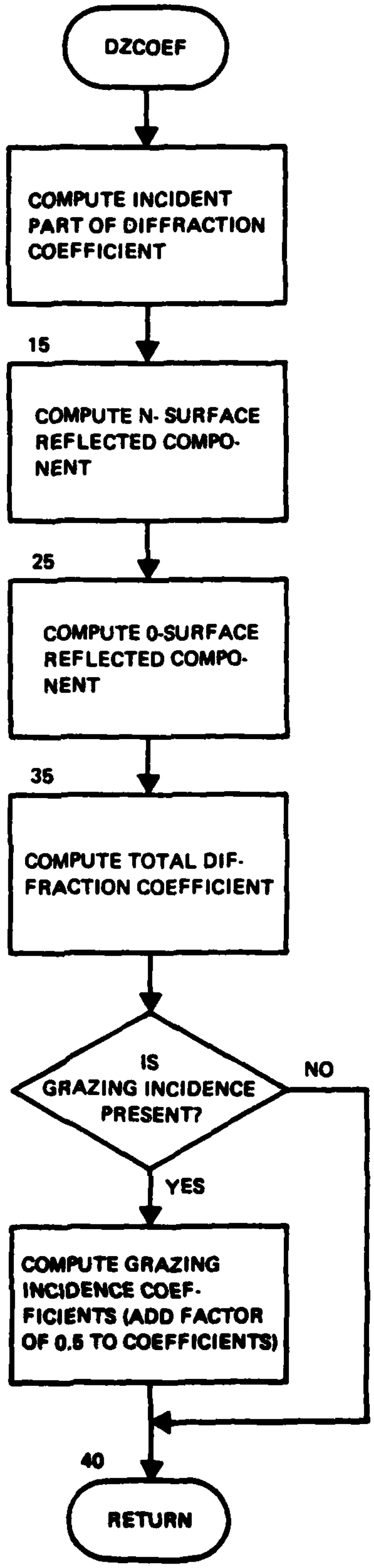
7. CALLED ROUTINES:

BEXP

FKY

8. REFERENCES:

A. R. G. Kouyoumjian and P. H. Pathak, "A Uniform Geometrical Theory of Diffraction for an Edge in a Perfectly Conducting Surface," Proc. IEEE, Vol. 62, November 1974, pp. 1448-1461.


DZCOEF
COMPUTE INCIDENT PART OF DIFFRACTION COEFFICIENT
15
COMPUTE N- SURFACE REFLECTED COMPO- NENT
25
COMPUTE 0-SURFACE REFLECTED COMPO- NENT
35
COMPUTE TOTAL DIF- FRACTION COEFFICIENT
IS GRAZING INCIDENCE PRESENT?
NO
YES
COMPUTE GRAZING INCIDENCE COEF- FICIENTS (ADD FACTOR OF 0.5 TO COEFFICIENTS)
40
RETURN

1. NAME: EFDGEO (INPUT)

2. PURPOSE: Find the geometry data set linked to the solution argument of the E-field command and return its index in INTARG.

3. METHOD: The index of the solution data set is determined by checking for the presence or absence of the field data set. The lineage of the solution data set is searched for a geometry data set. If none is found, a warning message is printed and an error flag set.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
IGEOBT	Data set geometry attribute flag
INDX	Pointer to data set linked to solution data set
INDXA	Pointer to field data set
INDXB	Pointer to solution data set
LINKB	Data set linked to data set pointed to by INDX
LNKBIT	Attribute word of linked data set
NDXARG	INTARG location of geometry data set pointer

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
INTARG	/ARGCOM/
IPASS	/ARGCOM/
ISON	/ADEBUG/
KBGEOM	/PARTAB/
KOLBIT	/PARTAB/
KOLLNK	/PARTAB/
NDATBL	/PARTAB/

NOPCOD	/ADEBUG/
NPDATA	/PARTAB/
NUMARG	/ARGCOM/
B. OUTPUT	LOCATION
INTARG	/ARGCOM/
NOGOFG	/SCNPAR/

6. CALLING ROUTINE:

   TSKXQT

7. CALLED ROUTINES:

   ASSIGN

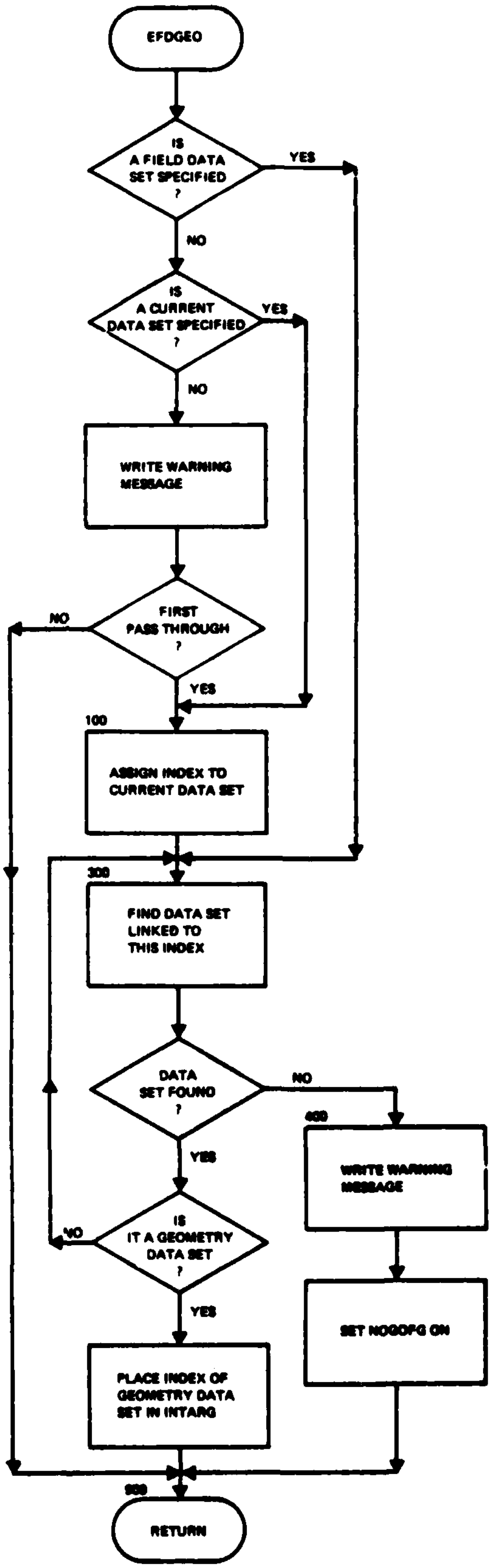
   IBITCK

   STATIN

   STATOT

   WLKBCK

EFDGEO (INPUT)



1. NAME: EGFMAT (MOM)

2. PURPOSE: Generate the scattered fields from the Green's function matrix and solution vector and add the result to the incident field to obtain the total field pattern.

3. METHOD: There are three data sets involved in the computation:

   (a) Solution vector, generated by MOM (S)
   (b) Green's function matrix, generated by GTD (G)
   (c) Field matrix, initially containing incident field data generated in GTD (F)

The formats of the matrices are shown in figure 1. In symbolic form, EGFMAT performs the operation

$$\underbrace{F}_{\substack{\text{total}\\\text{field}}} = \underbrace{F}_{\substack{\text{incident}\\\text{field}}} + \underbrace{G^T S}_{\substack{\text{scattered}\\\text{field}}} .$$

A row of G^T multiplied by S gives the scattered field for one field point. This value is added to the incident field (if any) for that point and stored as the total field. Hence several row-column multiplications are required to generate one column of F.

The array TEMP is used for in-core storage of columns of F, G, and S. To minimize file I/O, as many columns of G as possible are stored in TEMP simultaneously. The TEMP format is:

```
LOCS               | LOCG                                        | LOCFH   LOCF
 ↓                 | ↓                                           | ↓      ↙
[ The one column   | G(ICOLGX)  G(JCOLGX+1) | ... | ... | ...    | one column of ]
     of S          |                                             |   F(ICOLF)    
                     \_________________________________________/
```

KCOLG columns of G that contribute to this column of F
(as many as will fit)

The pointers LOCS, LOCG, LOCFH, and LOCF are the starting words of the solution vector, Green's function matrix, field matrix header, and field matrix data in TEMP.

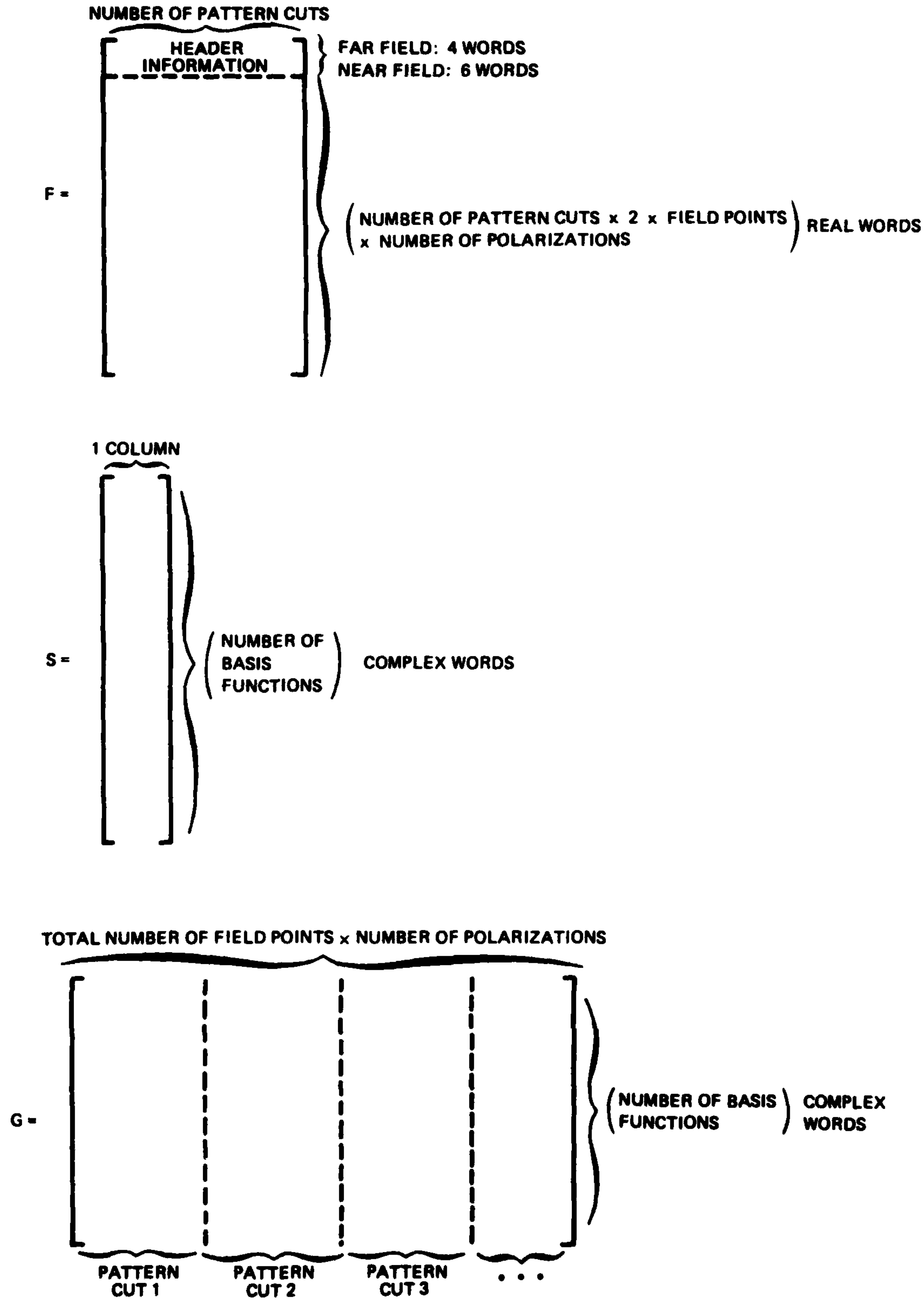


Figure 1. Formats of Data Sets Used by EGFMAT

There are four nested loops to EGFMAT:

(a) Loop over all columns of the field matrix (DO 200)
(b) Loop over the number of blocks of the Green's function matrix which will fit into TEMP (DO 150)
(c) Loop over Green's function matrix columns in TEMP (DO 140)
(d) Loop to multiply one column of Green's functon matrix by the solution vector (DO 130)

The inner most loop (d) generates the scattered field at one field point for one field polarization. Loops (b) and (c) span all field points and polarizations associated with a column of the field matrix. After execution of loop (b), this matrix column is stored and the next one retrieved. Loop (a) covers the entire field data set.

If the Green's function matrix is zero (no scattered field specified) only loop (a) is executed. This has the effect of advancing the edition of the field data set.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
ICOLF,ICOLFF	Column of field matrix in TEMP
ICOLGM	Column of Green's function matrix being multiplied by solution vector
ICOLGX	First column of Green's function matrix in TEMP
ICOLG1	First column of Green's function matrix associated with column of field matrix in TEMP
ICOLG2	Last column of Green's function matrix associated with column of field matrix in TEMP
IF	Pointer to real part of field element in TEMP being updated
IF1	Pointer to imaginary part of field element in TEMP being updated
IGF	Pointer to first element in Green's function matrix column being multiplied by solution vector

IGFM	Internal GEMACS literal code for "GFM"
ILIM	Last column of Green's function matrix in a block of columns
ILOW	First column of Green's function matrix in a block of diagrams
INDXF	Pointer to symbol table data for field matrix
INDXG	Pointer to symbol table data for Green's function matrix
INDXS	Pointer to symbol table data for solution vector
IOFFST	The relative location of the real part of the Green's function matrix element being multiplied by a solution vector element of the same relative location
IOFF1	The relative location of the imaginary part of the Green's function matrix element being multiplied by a solution vector element of the same relative location
KCOLG	Number of columns of Green's function matrix which will fit in TEMP
KSOLN	Flag indicating that the data set pointed to by INDXS is a solution data set
LOCF	Pointer to the beginning of field matrix data in TEMP
LOCFH	Pointer to the beginning of field matrix header data in TEMP
LOCG	Pointer to the beginning of Green's function matrix data in TEMP
LOCS	Pointer to the beginning of solution vector data in TEMP
N	Pointer to symbol table entry
NAME	User-assigned name of data set pointed to by N

NAMEF	User-assigned name of field matrix
NAMEG	Name of Green's function matrix, created by replacing the three rightmost characters in NAMEF by the letters contained in IGFM
NAMES	User-assigned name of solution vector
NBITF	Attribute word for field matrix
NBITG	Attribute word for Green's function vector
NBITS	Attribute word for solution vector
NCOLF	Number of columns in field matrix
NCOLG	Number of columns in Green's function matrix
NCOLS	Number of columns in solution vector
NEEDF	Number of data words required to store one column of field matrix
NEEDG	Number of data words required to store one column of Green's function matrix
NEEDS	Number of data words required to store one column of solution vector
NF	Hollerith format name of field matrix
NG	Hollerith format name of Green's function matrix
NINC	Number of real data words required per field point in field matrix
NROWF	Number of real words per column of field matrix
NROWG	Number of complex elements per column of Green's function matrix
NROWS	Number of complex elements per column of solution vector
NS	Hollerith format name of solution vector

NSHIFT	Number of bits in lower three characters of a GEMACS user-assigned name
SI	Imaginary part of scattered field
SR	Real part of scattered field

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
INDXF	F.P.
INDXS	F.P.
ISON	/ADEBUG/
KBSOLN	/PARTAB/
KJGTD	/INTMAT/
KJMOM	/INTMAT/
KOLBIT	/PARTAB/
KOLCOL	/PARTAB/
KOLNAM	/PARTAB/
KOLROW	/PARTAB/
LUPRNT	/ADEBUG/
NBYTSZ	/ADEBUG/
NDATBL	/PARTAB/
NINC	F.P.
NPDATA	/PARTAB/
NTEMPS	/TEMPO1/
NUMGTD	/GTDDAT/

B. OUTPUT

OUTPUT	LOCATION
IERRF	/ADEBUG/
TEMP	/TEMPO1/

6. CALLING ROUTINE:

   FLDDRV

7. CALLED ROUTINES:

   ASSIGN

   CONVRT

   ERROR

   GETSYM

   IBITCK

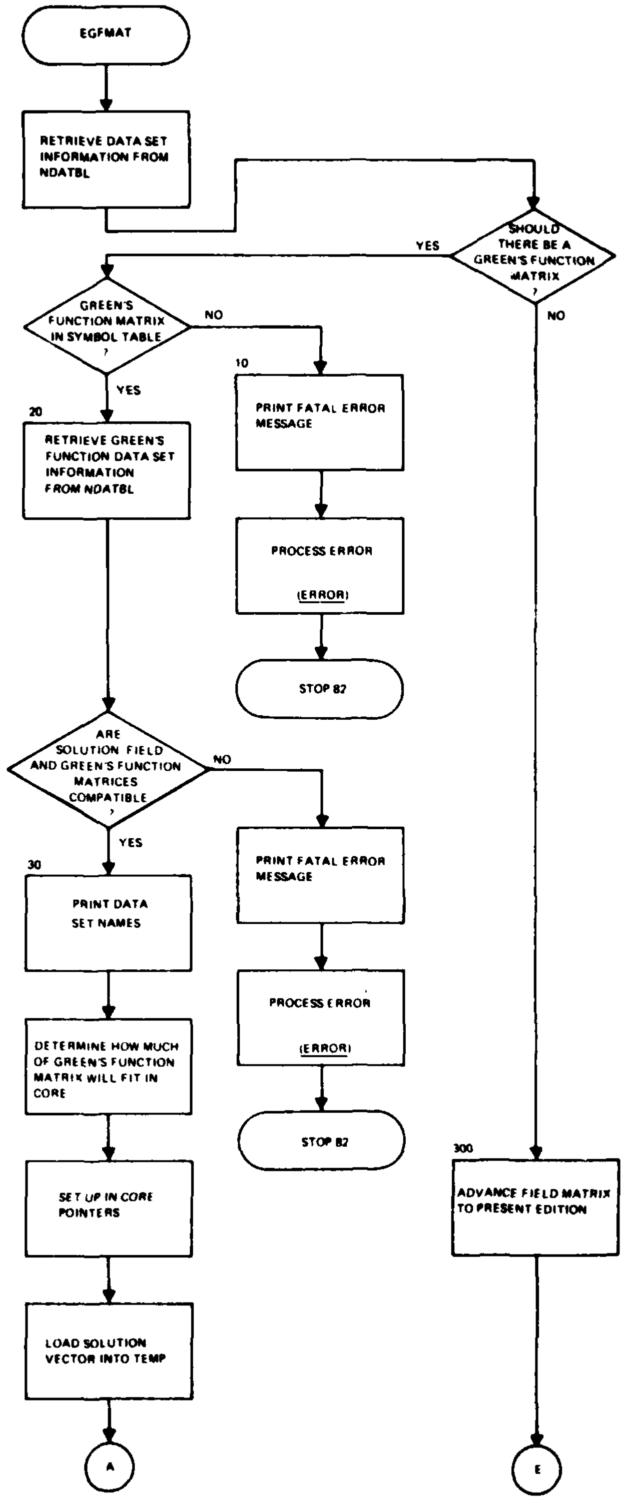
   PUTSYM

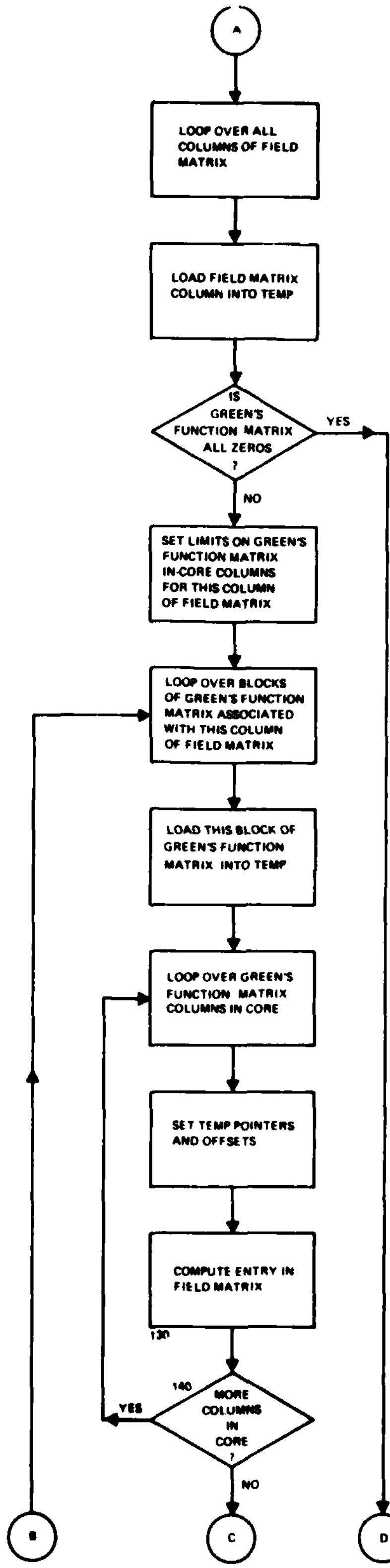
   STATIN

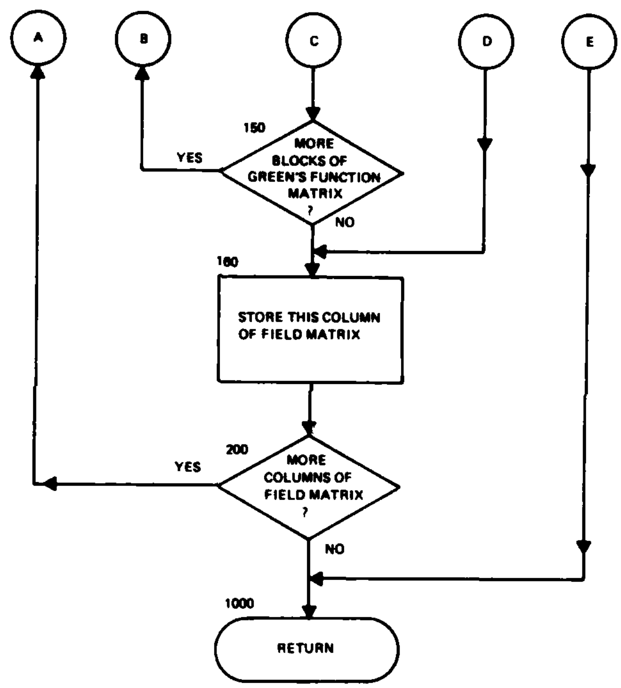
   STATOT

   SYMDEF

   WLKBCK


EGFMAT
RETRIEVE DATA SET INFORMATION FROM NDATBL
SHOULD THERE BE A GREEN'S FUNCTION MATRIX ?
YES
NO
GREEN'S FUNCTION MATRIX IN SYMBOL TABLE ?
NO
YES
10
PRINT FATAL ERROR MESSAGE
PROCESS ERROR
(ERROR)
STOP 82
20
RETRIEVE GREEN'S FUNCTION DATA SET INFORMATION FROM NDATBL
ARE SOLUTION FIELD AND GREEN'S FUNCTION MATRICES COMPATIBLE ?
NO
YES
30
PRINT DATA SET NAMES
PRINT FATAL ERROR MESSAGE
PROCESS ERROR
(ERROR)
STOP 82
DETERMINE HOW MUCH OF GREEN'S FUNCTION MATRIX WILL FIT IN CORE
SET UP IN CORE POINTERS
LOAD SOLUTION VECTOR INTO TEMP
A
300
ADVANCE FIELD MATRIX TO PRESENT EDITION
E


A
LOOP OVER ALL COLUMNS OF FIELD MATRIX
LOAD FIELD MATRIX COLUMN INTO TEMP
IS GREEN'S FUNCTION MATRIX ALL ZEROS ?
YES
NO
SET LIMITS ON GREEN'S FUNCTION MATRIX IN-CORE COLUMNS FOR THIS COLUMN OF FIELD MATRIX
LOOP OVER BLOCKS OF GREEN'S FUNCTION MATRIX ASSOCIATED WITH THIS COLUMN OF FIELD MATRIX
LOAD THIS BLOCK OF GREEN'S FUNCTION MATRIX INTO TEMP
LOOP OVER GREEN'S FUNCTION MATRIX COLUMNS IN CORE
SET TEMP POINTERS AND OFFSETS
COMPUTE ENTRY IN FIELD MATRIX
130
140
MORE COLUMNS IN CORE ?
YES
NO
B
C
D


A
B
C
D
E
150
YES
MORE BLOCKS OF GREEN'S FUNCTION MATRIX ?
NO
160
STORE THIS COLUMN OF FIELD MATRIX
200
YES
MORE COLUMNS OF FIELD MATRIX ?
NO
1000
RETURN

NAME: ENDCAP (INPUT)

PURPOSE: To store raw data GTD end cap geometry data in the segment table and make an entry in the point table for later geometry linkage by LNKGTD.

METHOD: Subroutine WYRDRV calls ENDCAP whenever an EC command is encountered in the geometry data. ENDCAP interprets the command items as scanned by SCAN and extracts values for end cap number and (θ,ϕ) direction of the end cap normal vector. If an optional item is not present, the values from the last EC command are used. Optional items are initially set to zero.

The format of the EC command is

EC	nn	[±cyl]	[theta]	[phi]	
1	2	3	4	5	NVAL/VAL index
		1	2	3	IO/FO index
	NTINT	NTINT	NTFLPT	NTFLPT	argument type (ITPARG)

nn = user-assigned end cap number

cyl = cylinder to which end cap is attached: { + = top end; - = bottom end }

theta = polar angle of end cap normal (degrees)

phi = azimuth angle of end cap normal (degrees)

ENDCAP places the following values in SEGTBL and PNTTBL:

	SEGTBL	PNTTBL
1	ITAG/IS	0
2	0	±cyl
3	0	0
4	THETA (radians)	
5	PHI (radians)	
6	0	
7	0	
8	0	
9	0	
10	0	
11	nn	

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
ERRFLG	Internal flag to indicate command error (integer)
FO	Array containing real values of arguments
ICYLN	Number of cylinder to which end cap is attached
IPT	Packed word containing tag and end cap numbers; each occupy 16 bits of the word
ISG	End cap segment number (assigned by SEGTBL)
ITAG	End cap tag number
ITPARG	Array of variable types (real, integer) for argument field
IO	Array containing integer values of arguments
NSGTBL	Number of SEGTBL entries required for ENDCAP geometry
NUMEC	User-assigned end cap number
THETA,PHI	Angles defining end cap normal (degrees)
THETA1,PHI1	Angles defining end cap normal (radians)

5. I/O VARIABLES:

A. INPUT	LOCATION
DGTORD	/GEODAT/
IECTAG	/GTDDAT/
IP217	/GEODAT/
ISOFF	/ADEBUG/
ISON	/ADEBUG/
LUPRNT	/ADEBUG/

MXECAR	/GTDDAT/
NARGS	/SCNPAR/
NCODE	/SCNPAR/
NTFLPT	/ADEBUG/
NTINT	/ADEBUG/
NVAL	/SCNPAR/
VAL	/SCNPAR/

B. OUTPUT LOCATION

NOGOFG	/SCNPAR/

6. CALLING ROUTINE:

WYRDRV

7. CALLED ROUTINES:

ASSIGN

PUTPNT

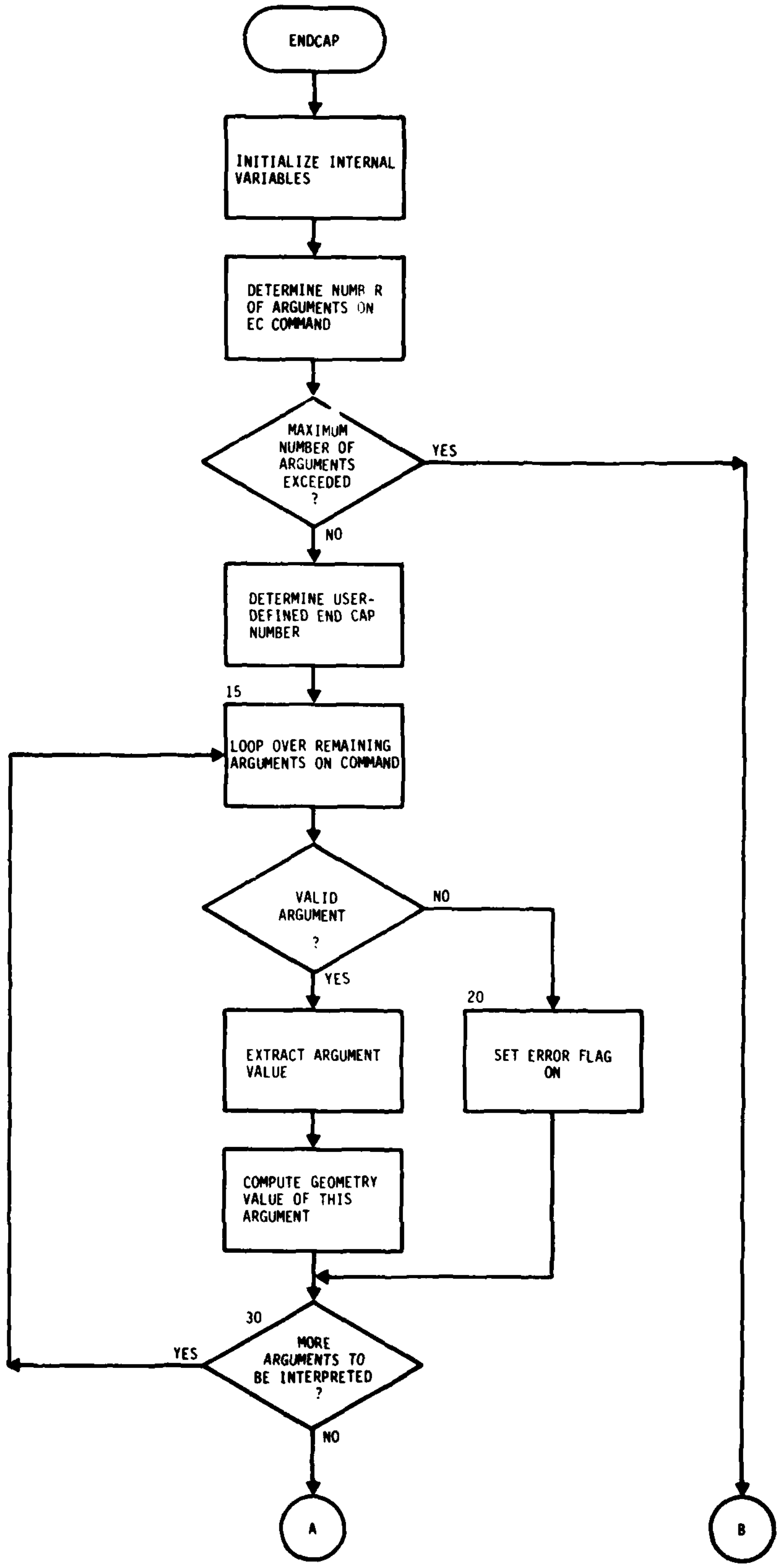
PUTSEG

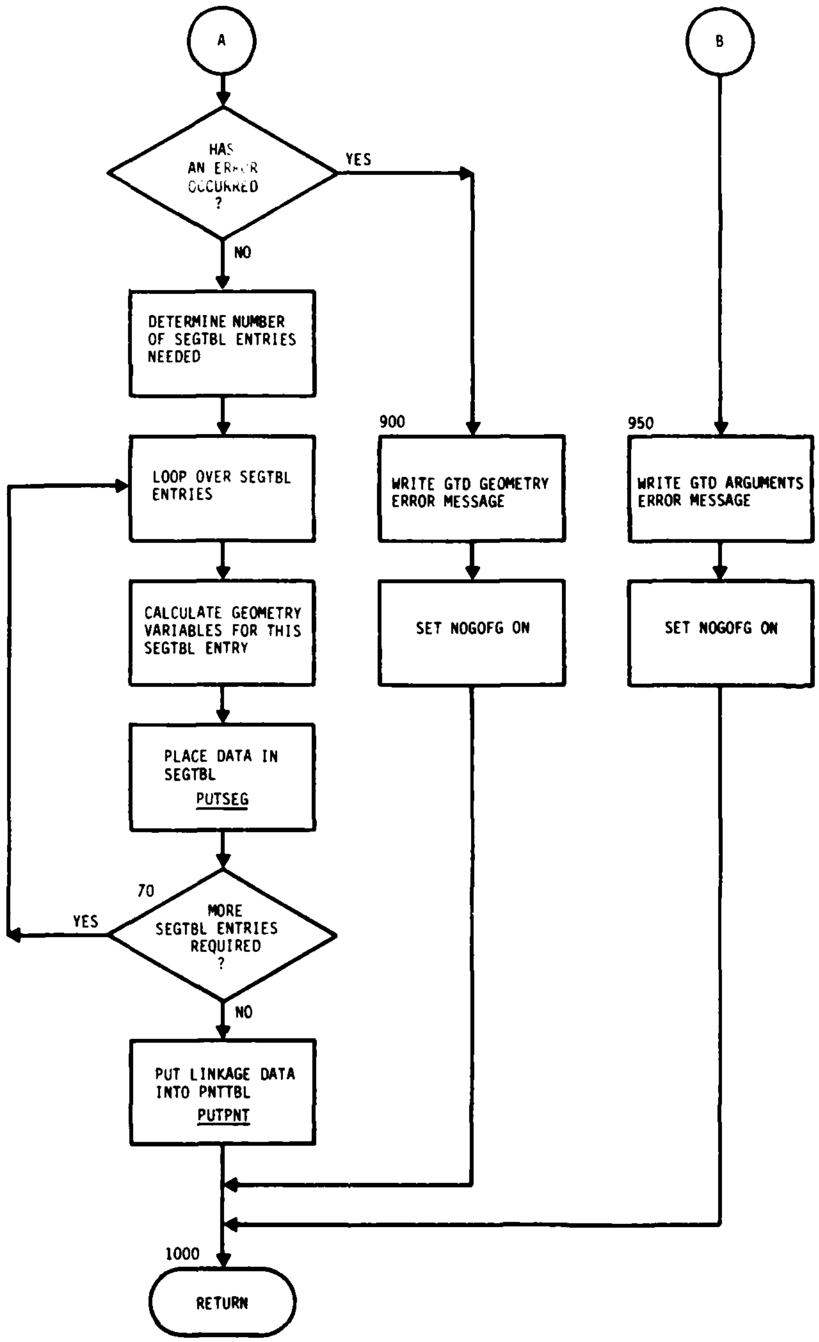
STATIN

STATOT

WLKBCK

ENDCAP (INPUT)




A
B
HAS AN ERROR OCCURRED ?
YES
NO
DETERMINE NUMBER OF SEGTBL ENTRIES NEEDED
LOOP OVER SEGTBL ENTRIES
CALCULATE GEOMETRY VARIABLES FOR THIS SEGTBL ENTRY
PLACE DATA IN SEGTBL
PUTSEG
70
MORE SEGTBL ENTRIES REQUIRED ?
YES
NO
PUT LINKAGE DATA INTO PNTTBL
PUTPNT
900
WRITE GTD GEOMETRY ERROR MESSAGE
SET NOGOFG ON
950
WRITE GTD ARGUMENTS ERROR MESSAGE
SET NOGOFG ON
1000
RETURN

1. NAME: ENDIF (GTD)

2. PURPOSE: To compute the unobstructed field due to diffraction off the elliptical cylinder end cap rim from a unit source in the given far-field direction or to a near-field observation point.

3. METHOD: ENDIF is the driver routine which directs all the ray tracing, physics and field calculations for end cap diffraction. The Uniform Geometrical Theory of Diffraction (see reference A) is used to compute the fields diffracted by the curved edges formed by the end cap disk and the curved surface of the elliptic cylinder. Details are given on pages 127-131 of reference B. The fields from four possible diffraction points on the edge are superimposed to give the total diffracted field from one end cap. For small regions of the radiation pattern, it is possible that three of the diffraction points will coalesce into one point leaving two diffraction points on the edge. When this happens a finite spike (pseudo-caustic) of small angular extent appears in the pattern. One way to correct for this is to use an equivalent current solution (see reference C). However, because this is costly in terms of computation time, it has not been included at present. The overall solution is not affected significantly by this approximation. Figure 1 shows the end cap diffraction point coordinate system which is used.

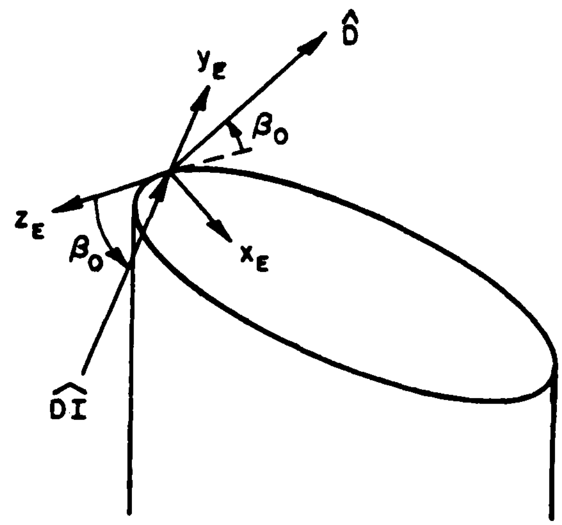


Figure 1. Illustration of Diffraction Point Coordinate System

The fields are first initialized to zero. Then the possible diffraction points are found in subroutine DFPTCL. The code then steps through the possible diffraction points and computes the fields for each point separately.

The ray path is checked for obstructions. If the path is shadowed, the present diffraction point is ignored and computations begin for the next point. If the ray path is clear, the source field pattern factor is found by calling subroutine SOURCE. The field incident at the diffraction point is determined. Then the end cap rim spreading radii, the phase factor, and the diffraction coefficient (from subroutine DZCOEF) are computed. The diffracted field perpendicular and parallel components are found as a function of these parameters and the incident field. The diffracted field is converted to theta and phi components in the reference coordinate system (RCS). Subroutine XYZFLD is then called to convert the field to x, y, z components and to accumulate it with fields from other interactions.

The code then goes to the next diffraction point and repeats the field computation process until all possible diffraction points have been considered. If the debug option is on, the total field due to end cap diffraction is printed on file LUPRNT. Control is then returned to the calling routine.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
AE	Radius of curvature of edge at diffraction point in end cap plane
CBO	Cosine of BO (dot product of diffracted ray and z axis of diffraction point coordinate system)
CPE	Cosine of PHER
CTE	Cosine of THER
CTHI	Dot product of incident ray propagation direction unit vector and cylinder unit normal
CV	Cosine of VR
D	X,Y,Z components of propagation direction after diffraction in RCS
DH	Diffraction coefficient for hard boundary condition

DHIT	Distance from source to nearest hit point (from subroutine PLAINT)
DI	X,Y,Z components of unit vector of incident ray propagation direction in RCS
DS	Diffraction coefficient for soft boundary condition
EDPH	Phi component of diffracted E-field in RCS
EDPHA	Field value amount of phi component to subtract off EDPHB so that EDPHB is the phi component for the end cap diffracted field from the diffraction point under consideration
EDPHB	Phi component of diffracted field in RCS from the diffraction point under consideration
EDPP	Component of diffracted field parallel to edge
EDPR	Component of diffracted field perpendicular to edge
EDTH	Theta component of diffracted E-field in RCS
EDTHA	Field value amount of theta component to subtract from EDTHB so that EDTHB is the theta component for the end cap diffracted field from the diffraction point under consideration
EDTHB	Theta component of diffracted field in RCS from the diffraction point under consideration
EF	Theta component of incident field pattern factor in RCS
EG	Phi component of incident field pattern factor in RCS
EIPP	Component of incident E-field parallel to edge

EIPR	Component of incident E-field perpendicular to edge
EIX,EIY,EIZ	X,Y,Z components of incident field pattern factor
EM	Normalization constant for z axis of diffraction point coordinate system
EX,EY,EZ	X,Y,Z components defining unit edge vector (z axis of diffraction point coordinate system)
FLDMAG	Field magnitude
FN	Wedge angle number
I	DO loop variable
LHIT	Set true if ray hits a plate (from subroutine PLAINT)
NC	End cap where diffraction occurs
NCC	Sign change variable
PH	Complex phase coefficient
PHEDR	Phi component of diffracted ray direction in diffraction point coordinate system
PHER	Phi component of incident ray propagation direction in diffraction point coordinate system
PHEX,PHEY,PHEZ	Polarization unit vector in phi direction for incident or diffracted ray in diffraction point coordinate system in x,y,z RCS components
PHIR	Phi component of incident ray direction in RCS
R	Parameter used in diffraction coefficient calculation
RG	Radius of curvature of cylinder surface at diffraction point in x-y plane

ENDIF (GTD)

RGAE	Radius of curvature of edge at diffraction point in end cap plane
RRN	Parameter used in diffraction coefficient calculation
SBO	Sine of BO
SNF	Distance between diffraction point and near-field observation point
SPE	Sine of PHER
SPM	Distance between diffraction point and source location
SPX,SPY,SPZ	X,Y,Z components of unit vector of propagation direction of incident ray
SSBO	Sine of BO squared
STE	Sine of THER
SV	Sine of VR
T1,T2,T3	X,Y,X components defining the incident (or diffracted) ray propagation direction in diffraction point coordinate system
THEDR	Theta component of diffracted ray direction in diffraction point coordinate system
THER	Theta component of incident ray propagation direction in diffraction point coordinate system
THEX,THEY,THEZ	Polarization unit vector in theta direction for incident or diffracted ray in diffraction point coordinate system in x,y,z RCS components
THIR	Theta component of incident ray direction in RCS
TOP	Computational variable
UB	X,Y,Z component of unit vector tangent to cylinder at diffraction point (2-D)
UN	X,Y,Z component of unit normal to cylinder at diffraction point (2-D)

UNEM	Normalization constant for edge unit normal NE
UNEX,UNEY,UNEZ	X,Y,Z components of unit normal to edge in end cap plane in RCS
V	Elliptical angles defining (up to) four diffraction points on end cap NC
VR	Elliptical angle defining diffraction point in RCS x-y plane
VXS	X,Y,Z components of unit vectors defining source coordinate system axes directions in RCS
XC	X,Y,Z components of diffraction point location in RCS
XEX,XEY,XEZ	X,Y,Z components defining unit vector of x axis of diffraction point coordinate system (vector normal to edge and parallel to end cap plane)
XSS	Source location
YEX,YEZ	X and Z components defining unit vector of y axis of diffraction point coordinate system (vector normal to end cap)

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
A	/GEOMEL/
B	/GEOMEL/
CJ	/COMP/
CNC	/GEOMEL/
CTC	/GEOMEL/
D	/DIR/
DP	/THPHUV/
DT	/THPHUV/

FLDPT	/NEAR/
LDEBUG	/TEST/
LNRFLD	/NEAR/
LUPRNT	/ADEBUG/
NC	F.P.
PHSR	/DIR/
PI	/PIS/
RPD	/PIS/
SNC	/GEOMEL/
THSR	/DIR/
TPI	/PIS/
VXS	/SORINF/
XS	/SORINF/
ZC	/GEOMEL/

B. OUTPUT LOCATION

EDPH	F.P.
EDTH	F.P.

6. CALLING ROUTINE:

GTDDRV

7. CALLED ROUTINES:

ASSIGN	NANDB	STATIN
BEXP	NFD	STATOT
BTAN2	PLAINT	TPNFLD
DFPTCL	SMAGNF	WLKBCK
DZCOEF	SOURCE	XYZFLD

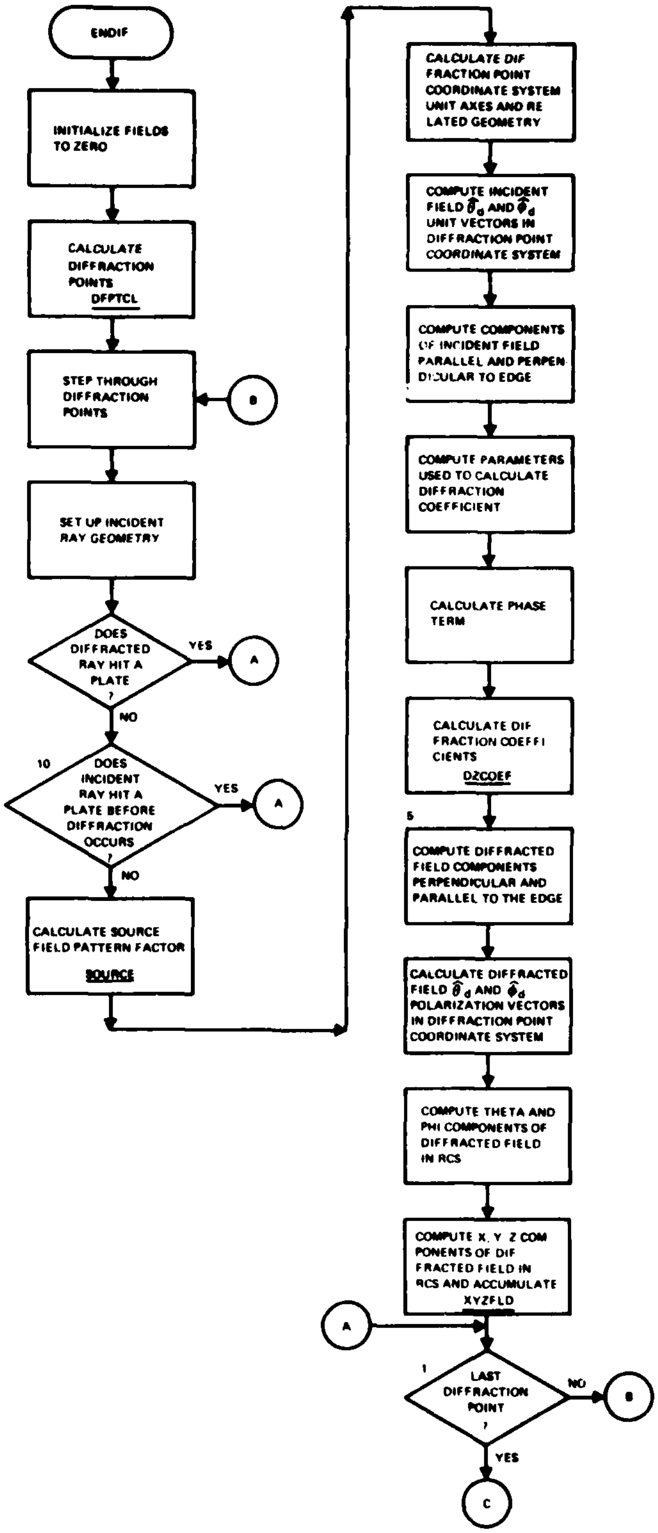
ENDIF (GTD)

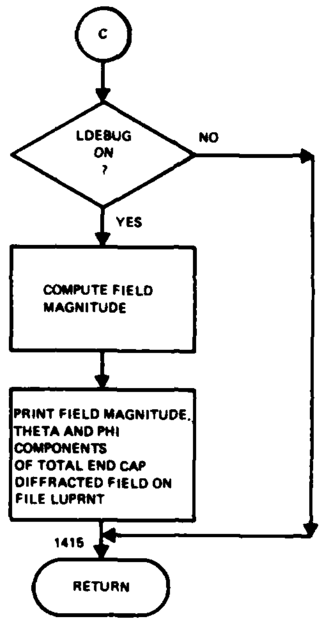
8. REFERENCES:

A. R. G. Kouyoumjian and P. H. Pathak, "A Uniform Geometrical Theory of Diffraction for an Edge in a Perfectly Conducting Surface," Proc. IEEE, Vol. 62, November 1974, pp. 1448-1461.

B. R. J. Marhefka, "Analysis of Aircraft Wing-Mounted Antenna Patterns," Report 2902-25, June 1976, The Ohio State University ElectroScience Laboratory, Department of Electrical Engineering; prepared under Grant No. NGL 36-008-138 for National Aeronautics and Space Administration.

C. E. D. Greer and W. B. Burnside, "High Frequency Near Field Scattering by an Elliptic Disk," Report 4583-1, December 1976, The Ohio State University ElectroScience Laboratory, Department of Electrical Engineering; prepared under Contract No. N62269-76-C-0554 for Naval Air Development Center.


ENDIF
INITIALIZE FIELDS TO ZERO
CALCULATE DIFFRACTION POINTS
DFPTCL
STEP THROUGH DIFFRACTION POINTS
B
SET UP INCIDENT RAY GEOMETRY
DOES DIFFRACTED RAY HIT A PLATE ?
YES
A
NO
10
DOES INCIDENT RAY HIT A PLATE BEFORE DIFFRACTION OCCURS ?
YES
A
NO
CALCULATE SOURCE FIELD PATTERN FACTOR
SOURCE
CALCULATE DIFFRACTION POINT COORDINATE SYSTEM UNIT AXES AND RELATED GEOMETRY
COMPUTE INCIDENT FIELD $\hat{\theta}_d$ AND $\hat{\phi}_d$ UNIT VECTORS IN DIFFRACTION POINT COORDINATE SYSTEM
COMPUTE COMPONENTS OF INCIDENT FIELD PARALLEL AND PERPENDICULAR TO EDGE
COMPUTE PARAMETERS USED TO CALCULATE DIFFRACTION COEFFICIENT
CALCULATE PHASE TERM
CALCULATE DIFFRACTION COEFFICIENTS
DZCOEF
5
COMPUTE DIFFRACTED FIELD COMPONENTS PERPENDICULAR AND PARALLEL TO THE EDGE
CALCULATE DIFFRACTED FIELD $\hat{\theta}_d$ AND $\hat{\phi}_d$ POLARIZATION VECTORS IN DIFFRACTION POINT COORDINATE SYSTEM
COMPUTE THETA AND PHI COMPONENTS OF DIFFRACTED FIELD IN RCS
COMPUTE X, Y, Z COMPONENTS OF DIFFRACTED FIELD IN RCS AND ACCUMULATE
XYZFLD
A
1
LAST DIFFRACTION POINT ?
NO
B
YES
C


C
LDEBUG ON ?
NO
YES
COMPUTE FIELD MAGNITUDE
PRINT FIELD MAGNITUDE, THETA AND PHI COMPONENTS OF TOTAL END CAP DIFFRACTED FIELD ON FILE LUPRNT
1415
RETURN

1. NAME: ERROR (GTD, INPUT, MOM, OUTPUT)

2. PURPOSE: Subroutine to initiate walk back and checkpoint in case of error termination.

3. METHOD: Not applicable.

4. INTERNAL VARIABLES: Not applicable.

5. I/O VARIABLES:

   A. INPUT LOCATION

INPUT	LOCATION
CHKPNT	/SYSFIL/
CHKWRT	/SYSFIL/
IOFILE	/IOFLES/

   B. OUTPUT:

   NONE

6. CALLING ROUTINES*:

   BACSUB (3)

   BANDIT (3)

   CNTGND (3)

   CNVGTD (1)

   COORDS (1)

   DECOMP (3)

   DMPDRV (1,2,3,4)

   EGFMAT (3)

   ESPARM (2)

   EXCDRV (2,3)

*1-INPUT
2-GTD
3-MOM
4-OUTPUT

ERROR (GTD, INPUT, MOM, OUTPUT)

FABL04 (3)

FLDDRV (2,3,4)

FLDOUT (4)

FNDREC (1,2,3,4)

GETARG (1,2,3,4)

GETKWV (1,2,3,4)

GETSYM (1,2,3,4)

LODDRV (3)

LUDDRV (3)

MOVFIL (1,2,3,4)

OPNFIL (1,2,3,4)

PLTDRV (1)

PRESCN (1)

PUTKWV (1,2,3,4)

PUTPNT (1)

PUTSYM (1,2,3,4)

RDEFIL (1,2,3,4)

RESTRT (1)

REBLCK (3)

SEJCON (2,3)

SETDRV (3)

SMATRX (3)

SOLDRV (3)

*1-INPUT
2-GTD
3-MOM
4-OUTPUT

SYMDEF (1,2,3,4)

SYMUPD (1,2,3,4)

SYSCHK (1,2,3,4)

TSKXQT (1,2,3,4)

WRTFIL (1,2,3,4)

ZGTDRV (2)

ZIJDRV (2,3)

ZIJSET (3)

7. CALLED ROUTINES:

CLSFIL

STATFN

STATIN

STATOT

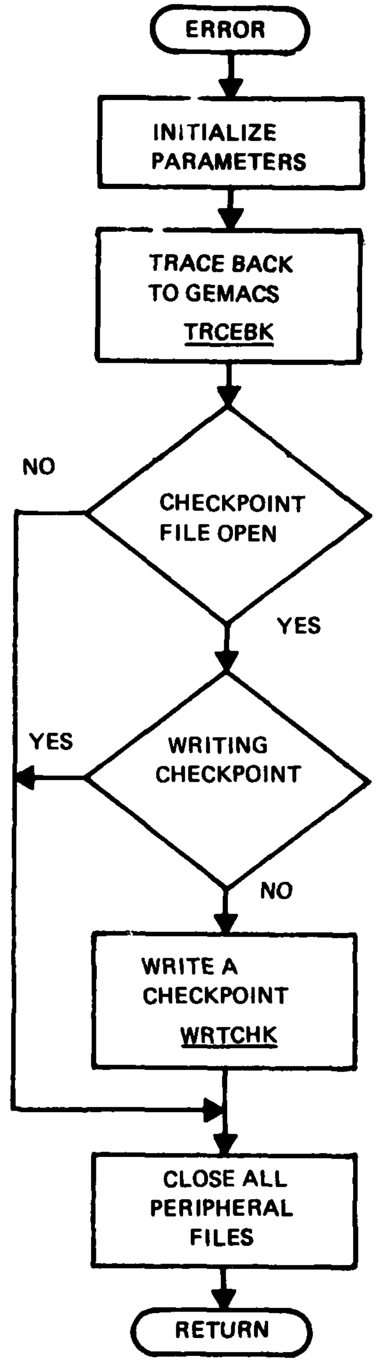
TRCEBK

WLKBCK

WRTCHK

*1-INPUT
2-GTD
3-MOM
4-OUTPUT

ERROR (GTD, INPUT, MOM, OUTPUT)



1. NAME: ESPARM (GTD)

2. PURPOSE: Obtain location and excitation of field sources (ESRC command) for use by ZGTDRV in computing fields incident on MOM geometries (excitation vector) or field patterns scattered by GTD geometries (field vector).

3. METHOD: ESPARM obtains source information by decoding NARGTB for excitation strength (VTHETA,VPHI), source type (IXTYPE), location (R,THETA,PHI), and eccentricity (ECC). The first call to ESPARM resets an internal task pointer (MTASK) to the present task plus 1. The task pointer is decremented and the task type is examined. A LOOP or LABEL task, a direct manipulation (DMP) task which changes the frequency, or a GMDATA task which changes the name of the geometry data set terminates the search up through the task table, and the routine is exited.

   A command other than ESRC causes ESPARM to decrement the task pointer and continue the search upward through NTSKTB.

   When an ESRC command is found, the excitation data set name and excited geometry data set name are checked. If either is not the same as requested, the task pointer is decremented and the search continues. If both ESRC data sets are correct, the ESRC command parameters are decoded from NARGTB, and the source location and excitation calculated.

   A subsequent call to ESPARM continues the search up the task table for the next proper ESRC command. All ESRC commands which contribute to a field value may be found by this method.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
COSETA	COS(ETAE)
COSP	COS(PHI)
COST	COS(THETA)
ECC	Source eccentricity
EM	Source excitation magnitude
EPX,EPY,EPZ	X,Y,Z components of cross-polarized source field

ESX,ESY,ESZ	X,Y,Z components of copolarized source field
ETAE	Polarization angle of source
ETI,EPI	θ,φ components of cross-polarized source field
ETR,EPR	θ,φ components of copolarized source field
ICW	IXTYPE value for cylinder wave
IDP	IXTYPE value for stiff dipole source
IPOL	Dipole polarization (1=x, 2=y, 3=z)
ISW	IXTYPE value for spherical wave
JCALL	Number of calls to ESPARM for this pattern
LOCGEO	Location of geometry data set name in NDATBL
LOCLIT	Location of literal value in LITNUM
LOCNAM	Location of excitation data set name in NDATBL
LOCTSK	Location of task in NARGTB
MTASK	Internal task table pointer
NAME	Name of excitation data set at task MTASK
NAMEG	Name of geometry data set at task MTASK
NAMGEO	Name of geometry data set presently being used
NDXARG	Pointer to task table
NFRQ	NCODES pointer to "FRQ"
NGEOM	NCODES pointer to default geometry name
NUMTSK	Task number of task pointed to by MTASK
PHI	Azimuthal location of source

R	Radial location of source
SINETA	SIN(ETAE)
SINP	SIN(PHI)
SINT	SIN(THETA)
THETA	Polar angle location of source
VPHI	$\hat{\phi}$ - component of source excitation
VTHETA	$\hat{\theta}$ - component of source excitation

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
DGTORD	/GEODAT/
FLTARG	/ARGCOM/
INTARG	/ARGCOM/
ISOFF	/ADEBUG/
ISON	/ADEBUG/
JCALL	F.P.
JTASK	/FLDVAL/
KOLCOD	/PARTAB/
KOLNAM	/PARTAB/
KOLVAL	/PARTAB/
KWNAME	/PARTAB/
LITNUM	/PARTAB/
NAMGEO	F.P.
NAMSRC	/FLDVAL/
NARGTB	/PARTAB/
NCODES	/PARTAB/

NDATBL	/PARTAB/
NOPCOD	/ADEBUG/
NTFLPT	/ADEBUG/
NTKEYW	/ADEBUG/
NTSKTB	/PARTAB/
ZERO	/ADEBUG/
B. OUTPUT	LOCATION
E	/FLDVAL/
IERRF	/ADEBUG/
ISRCE	/FLDVAL/
JCALL	F.P.
X	/FLDVAL/
Y	/FLDVAL/
Z	/FLDVAL/

6. CALLING ROUTINE:

   GETFLD

7. CALLED ROUTINES:

   ASSIGN

   ERROR

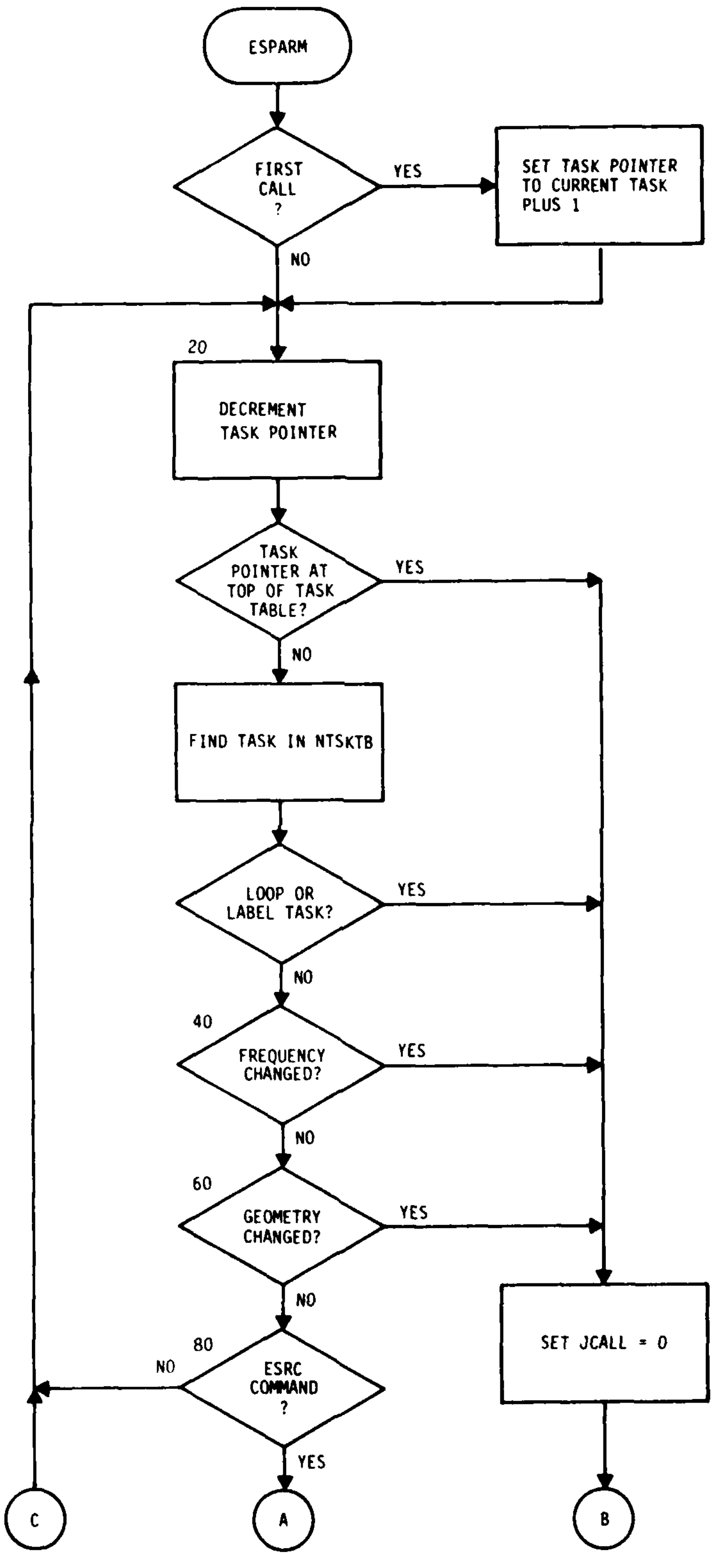
   GETARG

   STATIN

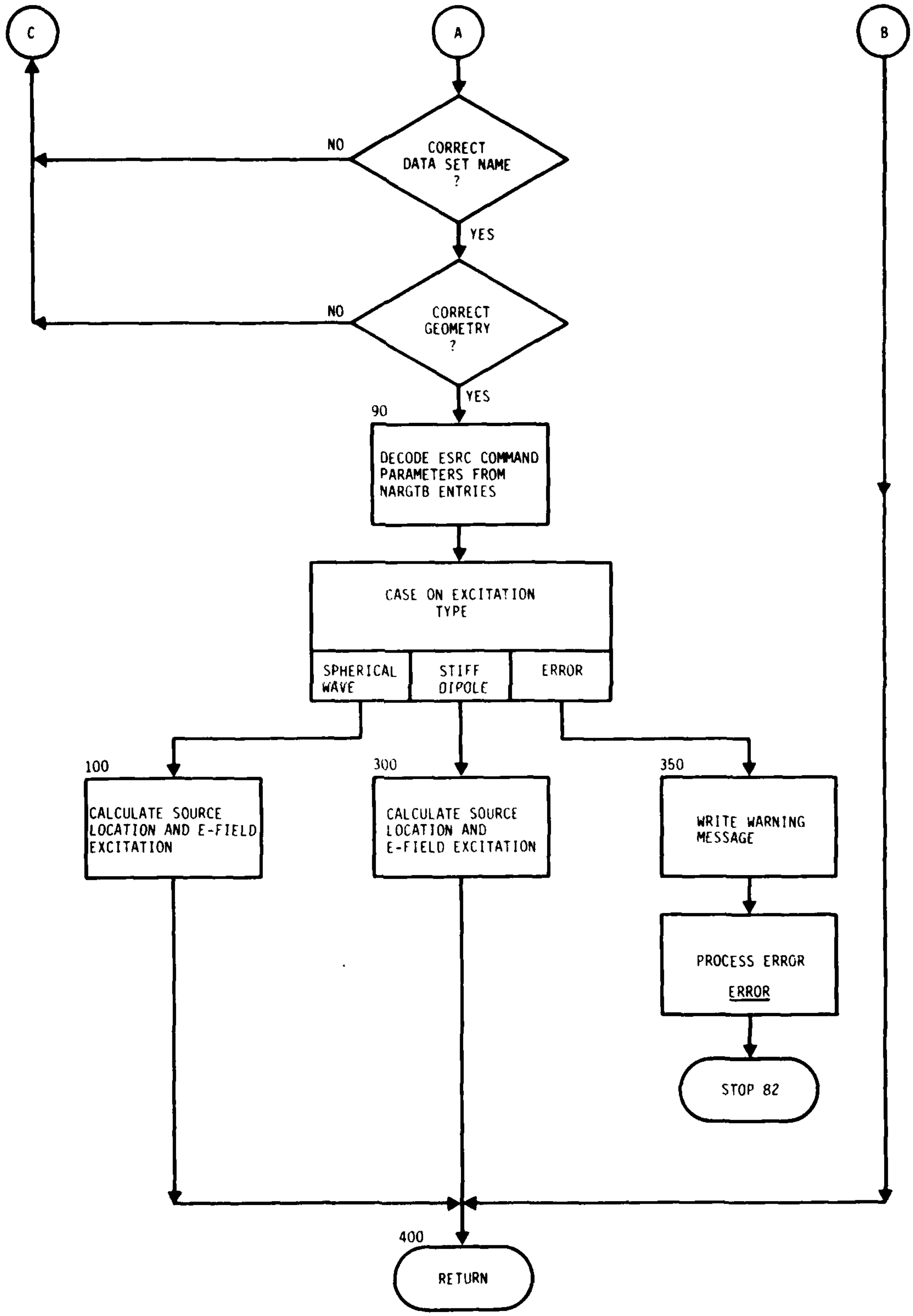
   STATOT

   WLKBCK

ESPARM (GTD)



ESPARM
FIRST CALL ?
YES
SET TASK POINTER TO CURRENT TASK PLUS 1
NO
20
DECREMENT TASK POINTER
TASK POINTER AT TOP OF TASK TABLE?
YES
NO
FIND TASK IN NTSKTB
LOOP OR LABEL TASK?
YES
NO
40
FREQUENCY CHANGED?
YES
NO
60
GEOMETRY CHANGED?
YES
NO
80
ESRC COMMAND ?
NO
YES
SET JCALL = 0
C
A
B


C
A
B
NO
CORRECT DATA SET NAME ?
YES
NO
CORRECT GEOMETRY ?
YES
90
DECODE ESRC COMMAND PARAMETERS FROM NARGTB ENTRIES
CASE ON EXCITATION TYPE
SPHERICAL WAVE
STIFF DIPOLE
ERROR
100
CALCULATE SOURCE LOCATION AND E-FIELD EXCITATION
300
CALCULATE SOURCE LOCATION AND E-FIELD EXCITATION
350
WRITE WARNING MESSAGE
PROCESS ERROR
ERROR
STOP 82
400
RETURN

1. NAME: EXCDRV (GTD)

2. PURPOSE: To calculate the GTD portion of fields incident on a MOM structure from a field source.

3. METHOD: EXCDRV examines the argument list for data sets of proper type. If no excitation data set is specified, the error flag is set and the routine is exited. If a geometry data set is not specified for the structure to be excited, an error message is printed and the error processed.

The contents of the geometry data set are examined next. If there are no MOM objects in the geometry, an excitation vector cannot be generated. In this case EXCDRV defines a null data set so that any subsequent EFIELD command can use the (null) data set name as input to compute incident fields.

An important feature of EXCDRV is its ability to add together excitations generated by different sources. If two (or more) sequential calls to EXCDRV are made with the same geometry data set name and at the same frequency, the excitation vectors are added. If this check shows any incompatibility between the last call to EXCDRV and the present call, the excitation data set is initialized to zero; otherwise, the data set is initialized with its previously calculated data.

Next, the interaction array is checked to see if any GTD interactions were specified. If not, or if there are no GTD objects in the geometry data set, the GTD contribution to the excitation vector is zero, and the routine exits.

The actual values of a field excitation are computed by a call to ZGTDRV. EXCDRV sets up the arguments to be passed to ZGTDRV, and ZGTDRV returns the excitation added into the TEMP array. Voltage excitation is not performed until MOM module execution.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
ECC	Eccentricity of source
FRQSAV	Excitation frequency default, saved from last call to EXCDRV
I	Loop index over GTD interactions
IBIT	Attribute word of excitation data set

ICW	Cylinder wave type identifier
IDP	Dipole source type identifier
IGEOM	Pointer to default geometry name in NCODES array
II	Number of GTD interactions
IOBS1,IOBS2	Limits on observation points
ISW	Spherical wave type identifier
ITYPE	Computation type: 2 = field source, geometry observation
IVS	Voltage source type identifier
IXCNAM	Source type label array
IXTYPE	Source type number
J	GTD interaction index
JSRC1,JSRC2	Limits on source points
K	GTD interaction index
KGBIT	Data set geometry flag
KJ	Array of GTD interactions
LNKEXC	Pointer to location of data set linked to excitation data set in symbol table
LOCEXC	Pointer to excitation data set location in symbol table
LOCYRS	Pointer to geometry data set location in symbol table
NAMEXC	User-defined name of excitation data set (internal format)
NAMYRS	User-defined name of geometry data set (internal format)
NBIT	Attribute word of null excitation data set
NC	Number of columns for excitation calculation

NCELLS	Number of words required for excitation calculation
NCOLE	Number of columns in previously defined excitation data set
NCOLS	Number of columns of null excitation data set
NDXARG	Pointer to excitation parameter in task table
NDXKWC	Pointer to cylinder wave keyword
NDXKWD	Pointer to dipole source keyword
NDXKWS	Pointer to spherical wave keyword
NR	Number of rows for excitation calculation
NROWE	Number of rows in previously defined excitation data set
NROWS	Number of rows in null excitation data set
NS	Name of geometry data set (A6 format)
NTASKE	Task number of ESRC command
NTASKV	Task number of VSRC command
NUMYRS	Column size of excitation matrix
NY	Name of geometry data set (A6 format)
VPHI	$\hat{\phi}$ - component of spherical wave excitation, imaginary part of dipole excitation
VTHETA	$\hat{\theta}$ - component of spherical wave, real part of dipole excitation

5. I/O VARIABLES:

A. INPUT	LOCATION
CLITE	/AMPZIJ/
FLTARG	/ARGCOM/

FRQMHZ	/AMPZIJ/
INTARG	/ARGCOM/
IPASS	/ARGCOM/
ISOFF	/ADEBUG/
ISON	/ADEBUG/
KBCPLX	/PARTAB/
KBGEOM	/PARTAB/
KBREAL	/PARTAB/
KBSRCE	/PARTAB/
KJGTD	/INTMAT/
KJINT	/INTMAT/
KOLBIT	/PARTAB/
KOLCOL	/PARTAB/
KOLLNK	/PARTAB/
KOLNAM	/PARTAB/
KOLROW	/PARTAB/
LSTARG	/ARGCOM/
LUPRNT	/ADEBUG/
NCODES	/PARTAB/
NDATBL	/PARTAB/
NOPCOD	/ADEBUG/
NPATCH	/SEGMNT/
NTEMPS	/TEMPO1/
NTFLPT	/ADEBUG/
NTSYMB	/ADEBUG/

NUMGTD	/GTDDAT/
NWIRE	/SEGMNT/
TEMP	/TEMP01/
TWOPI	/AMPZIJ/
ZERO	/ADEBUG/

B. OUTPUT — LOCATION

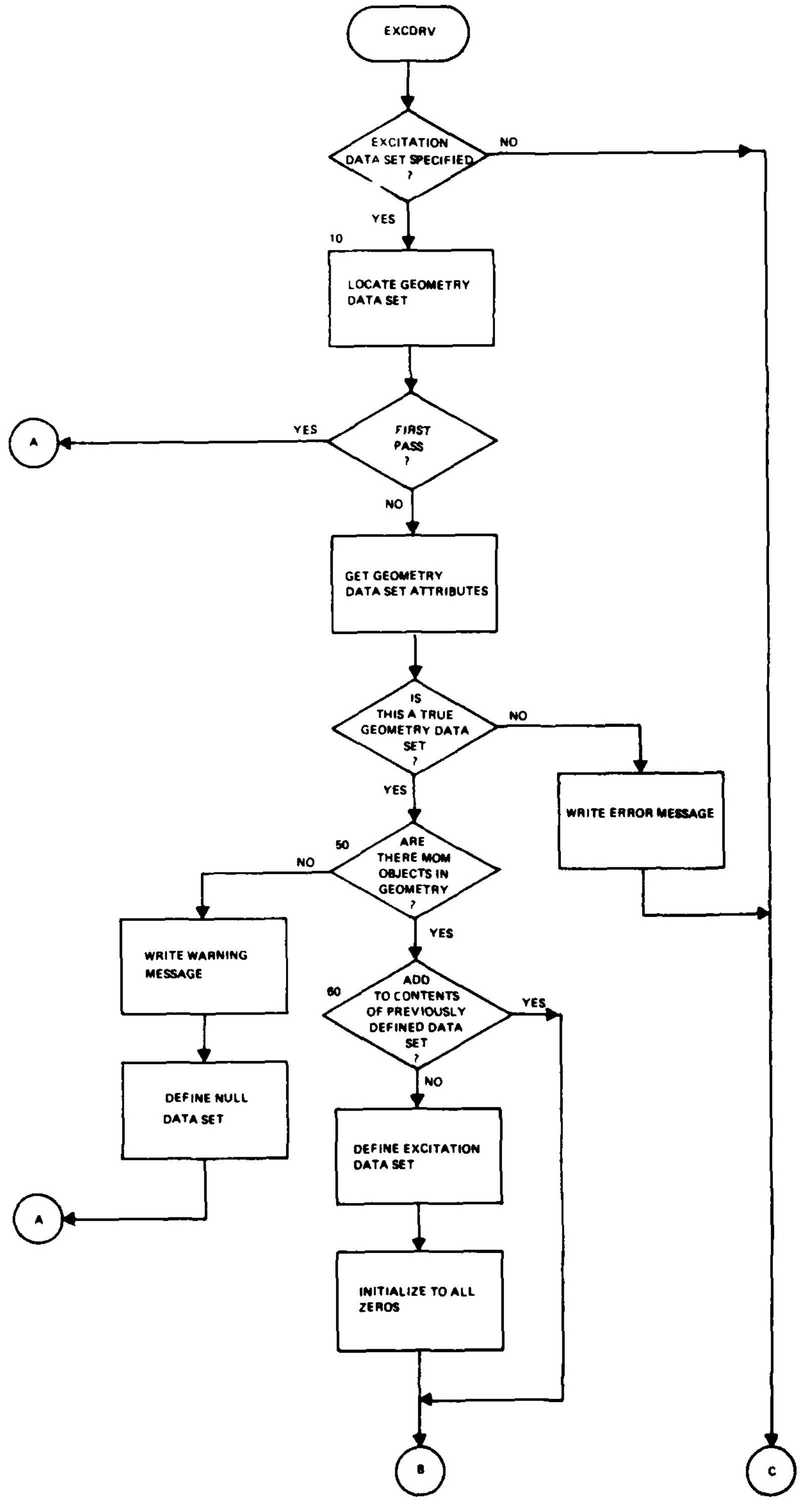
FRQMHZ	/AMPZIJ/
IERRF	/ADEBUG/
NAMSRC	/FLDVAL/
UPDBLK	/SEGMNT/
WAVLGH	/AMPZIJ/
WAVNUM	/AMPZIJ/

6. CALLING ROUTINE:

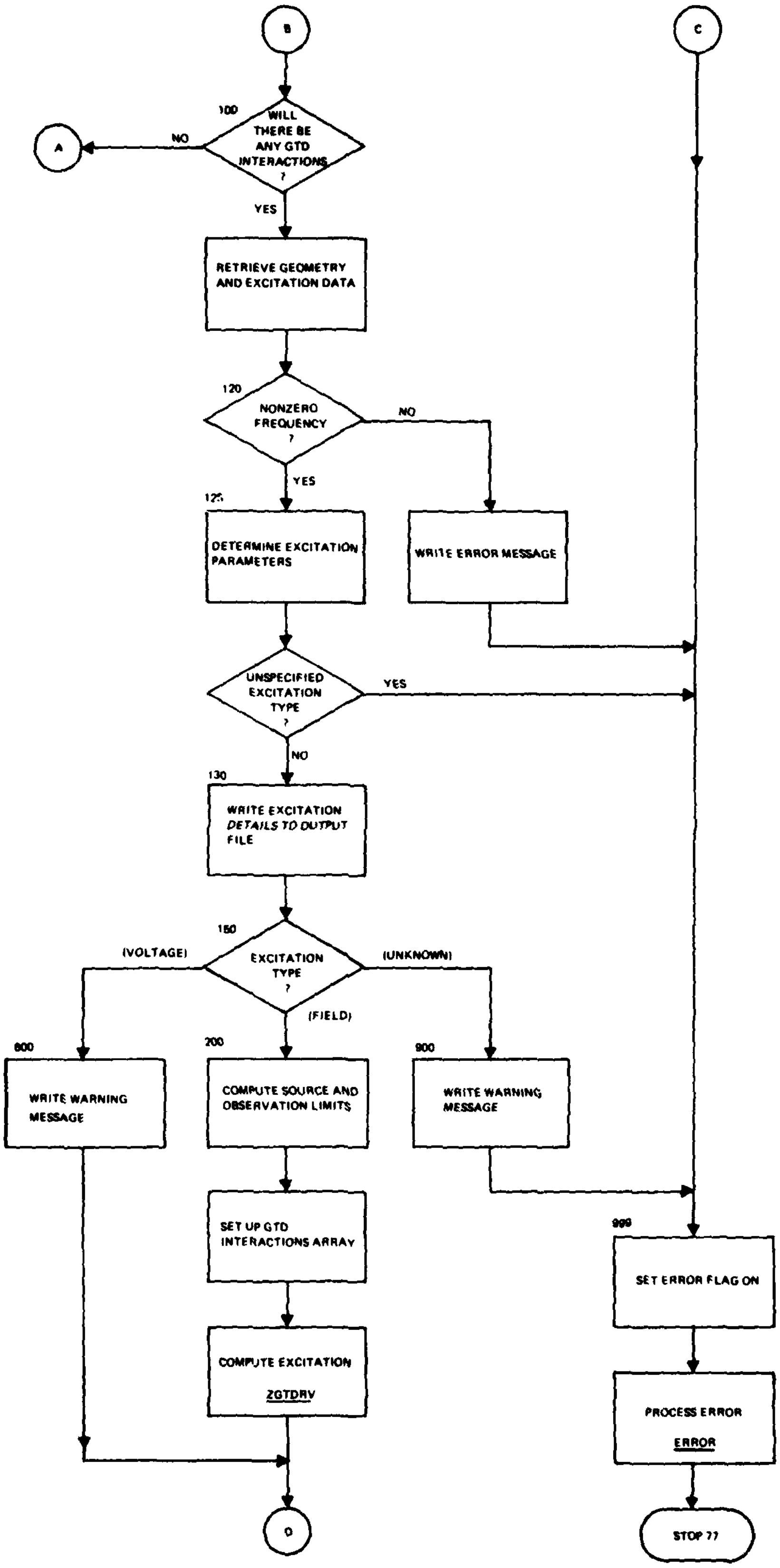
TSKXQT

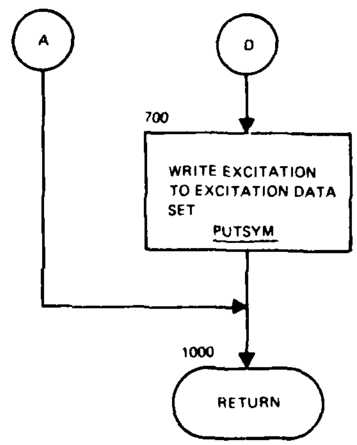
7. CALLED ROUTINES:

ASSIGN	PRTKJ
CONVRT	PUTSYM
ERROR	STATIN
GETARG	STATOT
GETGEO	SYMDEF
GETSEG	SYMUPD
GETSYM	WLKBCK
IBITCK	ZGTDRV


EXCDRV
EXCITATION DATA SET SPECIFIED ?
NO
YES
10
LOCATE GEOMETRY DATA SET
FIRST PASS ?
YES
A
NO
GET GEOMETRY DATA SET ATTRIBUTES
IS THIS A TRUE GEOMETRY DATA SET ?
NO
WRITE ERROR MESSAGE
YES
50
ARE THERE MOM OBJECTS IN GEOMETRY ?
NO
WRITE WARNING MESSAGE
DEFINE NULL DATA SET
A
YES
60
ADD TO CONTENTS OF PREVIOUSLY DEFINED DATA SET ?
YES
NO
DEFINE EXCITATION DATA SET
INITIALIZE TO ALL ZEROS
B
C

EXCDRV (GTD)


B
C
100
WILL THERE BE ANY GTD INTERACTIONS ?
NO
A
YES
RETRIEVE GEOMETRY AND EXCITATION DATA
120
NONZERO FREQUENCY ?
NO
YES
125
DETERMINE EXCITATION PARAMETERS
WRITE ERROR MESSAGE
UNSPECIFIED EXCITATION TYPE ?
YES
NO
130
WRITE EXCITATION DETAILS TO OUTPUT FILE
150
EXCITATION TYPE ?
(VOLTAGE)
(UNKNOWN)
(FIELD)
800
WRITE WARNING MESSAGE
200
COMPUTE SOURCE AND OBSERVATION LIMITS
900
WRITE WARNING MESSAGE
SET UP GTD INTERACTIONS ARRAY
999
SET ERROR FLAG ON
COMPUTE EXCITATION
ZGTDRV
PROCESS ERROR
ERROR
D
STOP 77


A
D
700
WRITE EXCITATION
TO EXCITATION DATA
SET
PUTSYM
1000
RETURN

1. NAME: EXCDRV (MOM)

2. PURPOSE: To calculate the MOM portions of fields incident on a MOM structure from a field source and/or the direct excitation from voltage sources.

3. METHOD: The argument list is retrieved and examined. Should there be no MOM objects in the geometry, an excitation vector cannot be defined. A warning message is printed and the routine exited.

   Since there is a possibility that GTD excitation data may already reside in the data set, care is taken not to overwrite these data but merely add to them. If the data set has been defined already, or if GTD data are in the data set, the data set is not re-initialized.

   The excitation type is checked. Valid excitations are spherical wave, plane wave, and voltage source. The first two types are wave type excitations, which are computed by subroutine SPWDRV. The voltage source excitations are generated internally according to the segments or tags specified by the VSRC command. The excitation is given by the voltage divided by the segment length, and a flag is set in the geometry data set to signal that the segment is excited with a voltage source. The data are stored in volts/meter.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
ECC	Eccentricity of plane wave or spherical wave source
ETAE	Angle of principal polarization component with respect to the θ direction (degrees)
FRQSAV	Excitation frequency default, saved from last call to EXCDRV
I	Loop index over voltage source excitation
IBIT	Attribute word of excitation data set
IBLK	Geometry data set block number
ICW	Cylinder wave type identifier
IDP	Dipole source type identifier
IGEOM	Pointer to default geometry name in NCODES array

ILOC1	Pointer to real part of excitation (one element) in TEMP
ILOC2	Pointer to imaginary part of excitation (one element) in TEMP
IS	Number of segment being excited
ISEG1	Beginning of segment list of voltage excitation
ISEG2	End of segment list for voltage excitation
ISGWRD	First ISGTBL word, containing tag and segment numbers
ISW	Spherical wave type identifier
ITAG1	Beginning of tag list for voltage excitation
ITAG2	End of tag list for voltage excitation
IVS	Voltage source type identifier
IXCNAM	Source type label array
IXTYPE	Source type number
JTAG	Tag identifier of segment
KGBIT	Data set geometry flag
LNKEXC	Pointer to location of data set linked to excitation data set in symbol table
LOCECC	Pointer to INTARG for ECC value
LOCEXC	Pointer to excitation data set location in symbol table
LOCYRS	Pointer to geometry data set location in symbol table
MDX	Keyword name of tag or segment identifier
N	Hollerith format of tag or segment identifier

NAMEXC	User-defined name of excitation data set (internal format)
NARGP1	NARGS + 1
NARGS	Number of arguments in excitation list
NCELLS	Number of words required for excitation calculation
NCOLE	Number of columns in previously defined excitation data set
NDX	Keyword number of tag or segment identifier
NDXARG	Pointer to excitation parameter in task table
NDXKWC	Pointer to cylinder wave keyword
NDXKWD	Pointer to dipole source keyword
NDXKWS	Pointer to spherical wave keyword
NE	Hollerith format of excitation type
NROWE	Number of rows in previously defined excitation data set
NS	Name of geometry data set (A6 format)
NTASKE	Task number of ESRC command
NTASKV	Task number of VSRC command
NUMYRS	Column size of excitation matrix
NXTARG	NARGS + 1
NY	Name of geometry data set (A6 format)
PHI	Aximuth angle of source location (degrees)
R	Radius of source location (meters)
SEGLGH	Length of excited segment (meters)
THETA	Polar angle of source location (degrees)

VMAG	Magnitude of field or voltage excitation
VPHI	$\hat{\phi}$-component of spherical wave excitation, imaginary part of voltage excitation
VTHETA	$\hat{\theta}$-component of spherical wave excitation, real part of voltage excitation

5. I/O VARIABLES

A. INPUT

INPUT	LOCATION
CLITE	/AMPZIJ/
DGTORD	/GEODAT/
FLTARG	/ARGCOM/
FRQMHZ	/AMPZIJ/
INTARG	/ARGCOM/
IP217	/GEODAT/
IPASS	/ARGCOM/
ISGTBL	/SEGMNT/
ISOFF	/ADEBUG/
ISON	/ADEBUG/
KBCPLX	/PARTAB/
KBGEOM	/PARTAB/
KBREAL	/PARTAB/
KBSRCE	/PARTAB/
KJGTD	/INTMAT/
KJMOM	/INTMAT/
KOLBIT	/PARTAB/
KOLCOL	/PARTAB/

EXCDRV (MOM)

KOLLNK	/PARTAB/
KOLNAM	/PARTAB/
KOLROW	/PARTAB/
KWNAME	/PARTAB/
KWSEGS	/PARTAB/
KWTAGS	/PARTAB/
LSTARG	/ARGCOM/
LUPRNT	/ADEBUG/
MAXBLK	/SEGMNT/
MAXSEG	/SEGMNT/
NCODES	/PARTAB/
NDATBL	/PARTAB/
NDXBLK	/SEGMNT/
NOPCOD	/ADEBUG/
NPATCH	/SEGMNT/
NTEMPS	/TEMP01/
NTFLPT	/ADEBUG/
NTSYMB	/ADEBUG/
NUMARG	/ARGCOM/
NUMSEG	/SEGMNT/
NWIRE	/SEGMNT/
SEGTBL	/SEGMNT/
TEMP	/TEMP01/
TWOPI	/AMPZIJ/
ZERO	/ADEBUG/

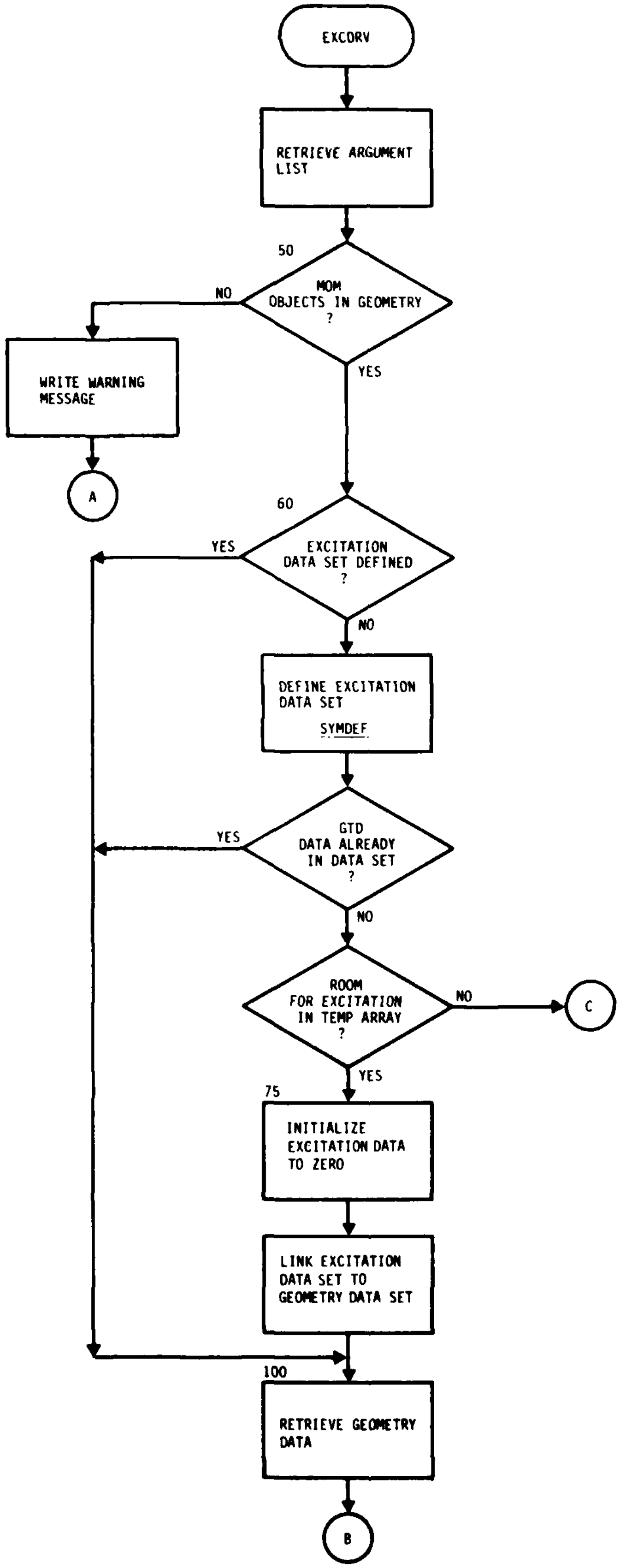
B. OUTPUT	LOCATION
FRQMHZ	/AMPZIJ/
IERRF	/ADEBUG/
NAMSRC	/FLDVAL/
SEGTBL	/SEGMNT/
TEMP	/T MP01/
UPDBLK	/SEGMNT/
WAVLGH	/AMPZIJ/
WAVNUM	/AMPZIJ/

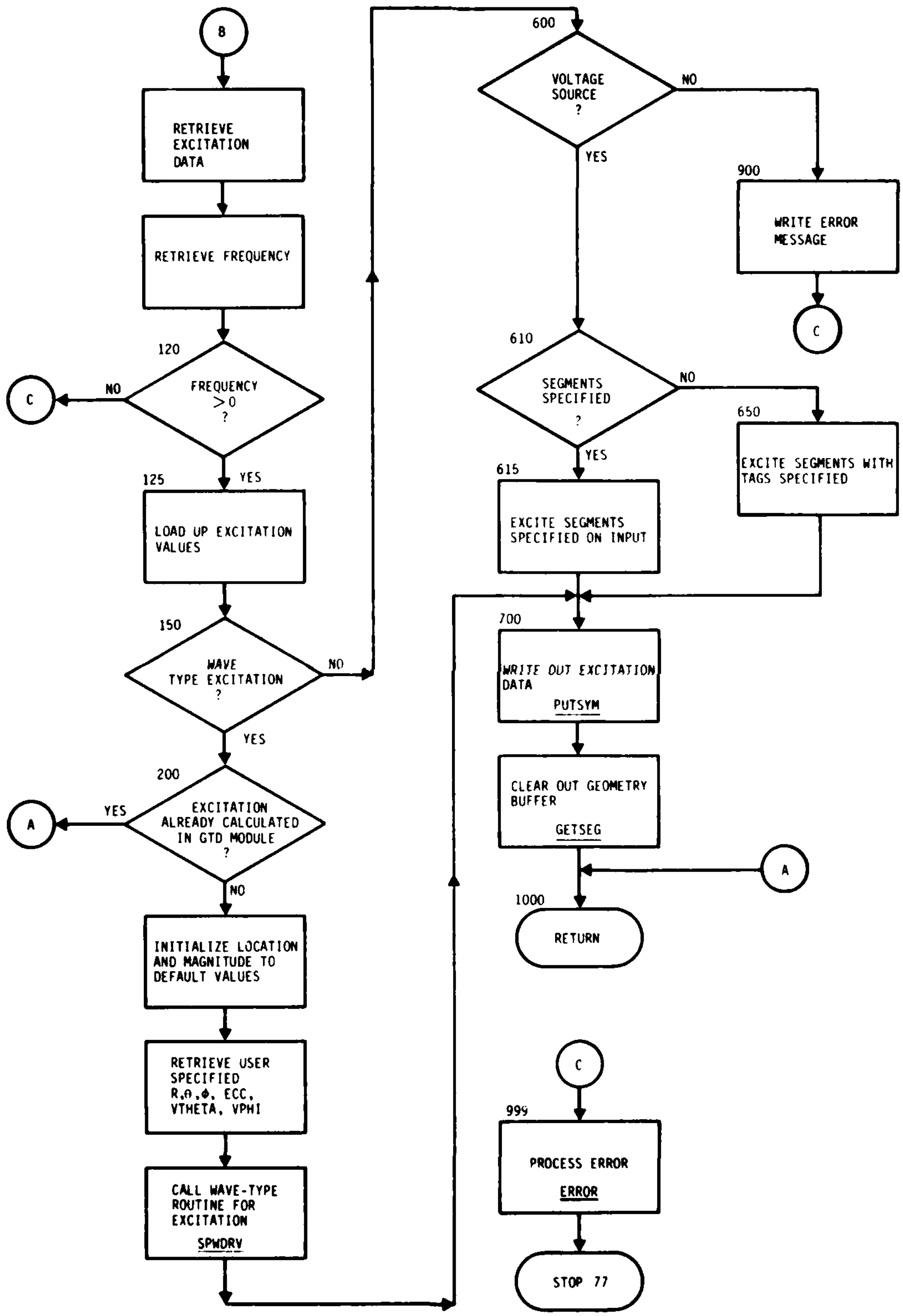
6. CALLING ROUTINE:

TSKXQT

7. CALLED ROUTINES:

ASSIGN	GETSEG	STATIN
CONVRT	GETSYM	STATOT
ERROR	IBITCK	SYMDEF
GETARG	PRTKJ	SYMUPD
GETGEO	PUTSYM	WLKBCK
	SPWDRV	ZZXDUM


EXCDRV
RETRIEVE ARGUMENT LIST
50
MOM OBJECTS IN GEOMETRY ?
NO
WRITE WARNING MESSAGE
A
YES
60
EXCITATION DATA SET DEFINED ?
YES
NO
DEFINE EXCITATION DATA SET
SYMDEF
GTD DATA ALREADY IN DATA SET ?
YES
NO
ROOM FOR EXCITATION IN TEMP ARRAY ?
NO
C
YES
75
INITIALIZE EXCITATION DATA TO ZERO
LINK EXCITATION DATA SET TO GEOMETRY DATA SET
100
RETRIEVE GEOMETRY DATA
B


B
RETRIEVE EXCITATION DATA
RETRIEVE FREQUENCY
120
FREQUENCY > 0 ?
NO
C
125
YES
LOAD UP EXCITATION VALUES
150
WAVE TYPE EXCITATION ?
NO
YES
200
EXCITATION ALREADY CALCULATED IN GTD MODULE ?
YES
A
NO
INITIALIZE LOCATION AND MAGNITUDE TO DEFAULT VALUES
RETRIEVE USER SPECIFIED R, θ, φ, ECC, VTHETA, VPHI
CALL WAVE-TYPE ROUTINE FOR EXCITATION
SPWDRV
600
VOLTAGE SOURCE ?
NO
YES
900
WRITE ERROR MESSAGE
C
610
SEGMENTS SPECIFIED ?
NO
650
EXCITE SEGMENTS WITH TAGS SPECIFIED
YES
615
EXCITE SEGMENTS SPECIFIED ON INPUT
700
WRITE OUT EXCITATION DATA
PUTSYM
CLEAR OUT GEOMETRY BUFFER
GETSEG
A
1000
RETURN
C
999
PROCESS ERROR
ERROR
STOP 77

1. NAME: FABL02 (INPUT)

2. PURPOSE: Contains output for Input Language Processor (ILP) routines.

3. METHOD: FABL02 receives an error number and prints out an error message for this number.

4. INTERNAL VARIABLES:

ERRMSG	Two-dimensional array containing error messages
IWRDS3	Error number
NPRMSG	Number of words for each message

5. I/O VARIABLES:

A. INPUT	LOCATION
IWORDS	/ADEBUG/
LUPRNT	/ADEBUG/

6. CALLING ROUTINES:

FNDARG

LITSCH

PARSE

PLIST

SYMLIT

SYMSCH

7. CALLED ROUTINES:

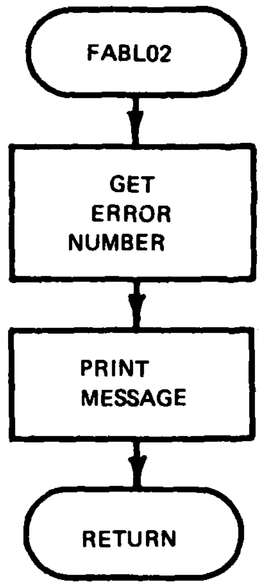
ASSIGN

STATIN

STATOT

WLKBCK

FABL02 (INPUT)



1. NAME: FABL04 (MOM)

2. PURPOSE: Subroutine FABL04 interfaces with the PRINT command to write out requested information.

3. METHOD: Requests for printing stored values of variables are interfaced through the PRINT command and routine PRTSYM. The routine is functionally divided into three areas to provide specific printing capabilities related to the following categories of requests:

   (a) Print a single variable
   (b) Print a single row or column variable
   (c) Print a matrix variable

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
DATTYP	Array containing data types
FMT	The format statement is built in this array
FMTFLD	Array containing variable format for output data
ICASE	Array containing format size
INCALL	Call number for the type of output
ITYPE	Type of data to be printed
NDXFLD	Index to the field type
NPRFMT	Size of the FMT array

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
DBGPRT	/ADEBUG/
INCALL	F.P.
IWORDS	/ADEBUG/
NUMWRD	/ADEBUG/
WORDS	/ADEBUG/

B. OUTPUT: NONE

6. CALLING ROUTINE:

   PRTSYM

7. CALLED ROUTINES:

   ASSIGN

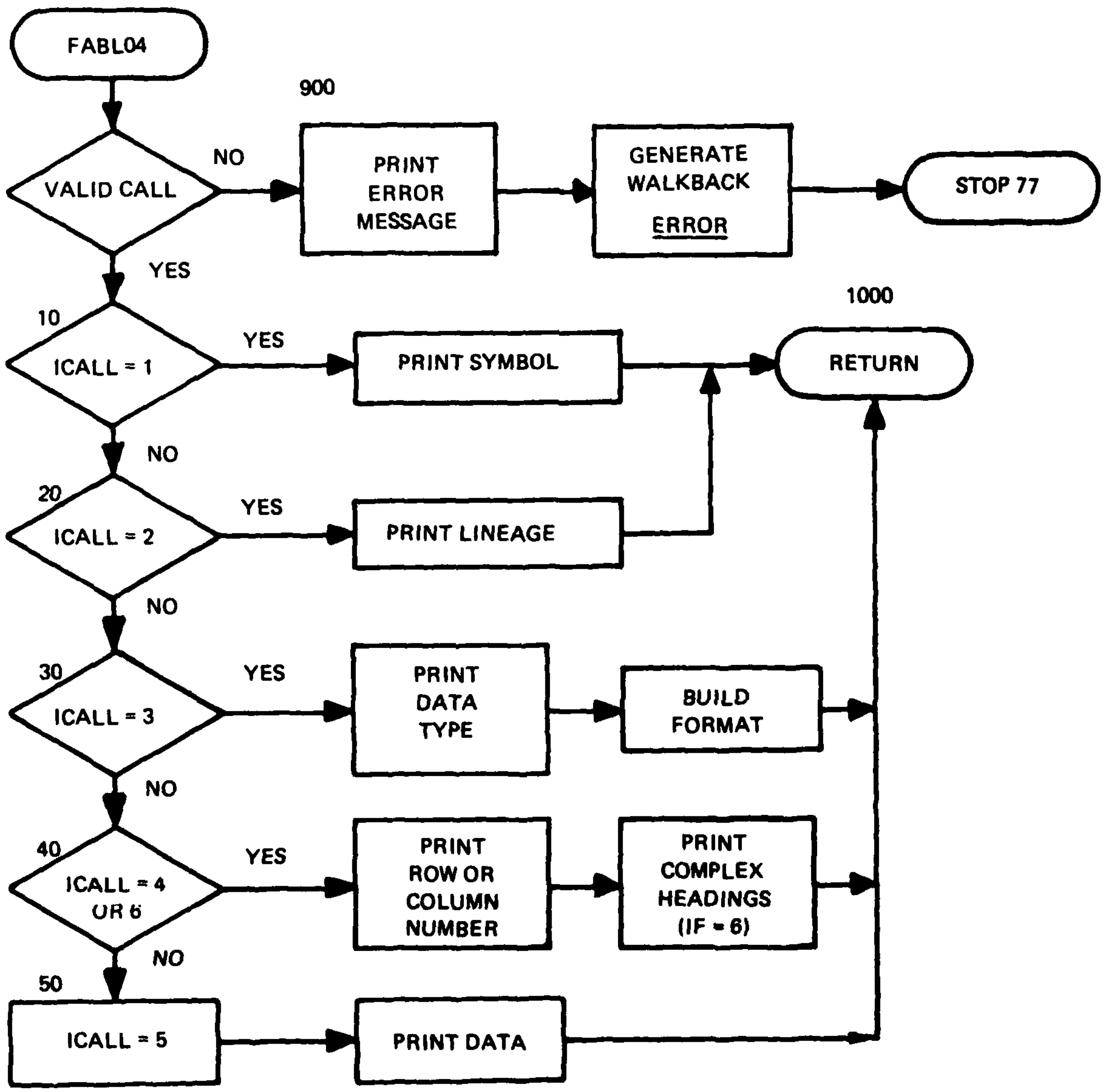
   ERROR

   STATIN

   STATOT

   WLKBCK

FABL04 (MOM)


FABL04
900
VALID CALL
NO
PRINT ERROR MESSAGE
GENERATE WALKBACK ERROR
STOP 77
YES
10
ICALL = 1
YES
PRINT SYMBOL
1000
RETURN
NO
20
ICALL = 2
YES
PRINT LINEAGE
NO
30
ICALL = 3
YES
PRINT DATA TYPE
BUILD FORMAT
NO
40
ICALL = 4 OR 6
YES
PRINT ROW OR COLUMN NUMBER
PRINT COMPLEX HEADINGS (IF = 6)
NO
50
ICALL = 5
PRINT DATA

1. NAME: FARFLD (MOM)

2. PURPOSE: To calculate the far electric field $\left(\text{neglecting } \frac{e^{-jkr}}{r}\right)$ in free space or over various types of grounds, where the ground effects are included by means of the Fresnel reflection coefficients.

3. METHOD: The far electric field due to line currents can be written

$$\bar{E}(\bar{r}) = j\omega\mu_o \frac{e^{-jkr}}{4\pi r} \left[\left(\hat{k} \, (\hat{k} \cdot \int e^{j\hat{k} \cdot \bar{r}'} \bar{I}(\bar{r}') \, d\ell) \right) - \int e^{j\hat{k} \cdot \bar{r}'} \bar{I}(\bar{r}') \, d\ell \right]$$

where $\bar{r}$ is the position vector of the observation point, $\bar{r}'$ is the position vector of the source point, $\bar{k}$ is in the direction of propagation with a magnitude of $2\pi/\lambda$. Specialized to straight wire segments, as used in the GEMACS formulation,

$$\bar{E}(\bar{r}) = jk \frac{\eta_o}{4\pi} \frac{e^{-jkr}}{r} \sum_{i=1}^{N} e^{j\bar{k} \cdot \bar{R}_i} \left[\hat{k} \, (\hat{k} \cdot \bar{Q}_i) - \bar{Q}_i \right] \quad \text{(Eq. 1)}$$

where η_o is free space impedance, $\bar{R}_i$ is the position vector of the center of the i^{th} segment and

$$\bar{Q}_i = \hat{u}_i \int_{-(s/2)}^{s/2} e^{j2\pi(\hat{k} \cdot \hat{u}_i)t} \left(\frac{I_i(t)}{\lambda} \right) \lambda dt$$

where $\hat{u}_i = \cos\alpha_i \cos\beta_i \, \hat{x} + \cos\alpha_i \sin\beta_i \, \hat{y} + \sin\alpha_i \, \hat{z}$, which is the reference direction of the i^{th} segment with the angles defined as shown in figure 1, $\lambda t\hat{u}_i = \bar{r}' - \bar{R}_i$, and s is the segment length.

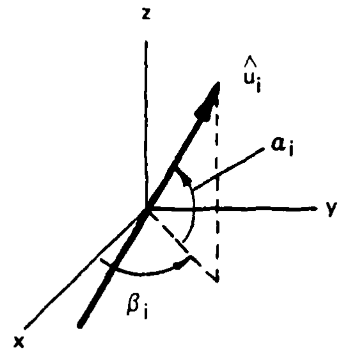


Figure 1. Segment Orientation

With

$$I_i(t)/\lambda = A_i + B_i \sin 2\pi t + C_i \cos 2\pi t$$

integration of $\bar{Q}_i$ yields

$$\bar{Q}_i = \hat{u}_i \left[A_i \frac{\sin \pi w_i s}{\pi w_i} + j B_i \left(\frac{\sin \pi(1 + w_i)s}{2\pi(1 + w_i)} - \frac{\sin \pi(1 - w_i)s}{2\pi(1 - w_i)} \right) \right.$$

$$\left. + C_i \left(\frac{\sin \pi(1 + w_i)s}{2\pi(1 + w_i)} + \frac{\sin \pi(1 - w_i)s}{2\pi(1 - w_i)} \right) \right] \qquad \text{(Eq. 2)}$$

where

$$w_i = -\hat{k} \cdot \hat{u}_i \qquad \text{(Eq. 3)}$$

Note,theterm$\hat{k}$ ($\hat{k} \cdot \bar{Q}_i$ in equation (1) is completely radial and cancels the radial component of $\bar{Q}_i$. This term is ignored in FARFLD since the desired transverse components will be computed by a dot product. Thus, for program use only and with the understanding that only tranverse components will be used, we write

$$\bar{E}(\bar{r}) = -j \frac{\eta_o k}{4\pi} \frac{e^{-jkr}}{r} \sum_{i-1}^{N} e^{j\bar{k} \cdot \bar{R}_i} \bar{Q}_i \qquad \text{(Eq. 4)}$$

The far electric field at a location $\bar{r}$ due to a surface current density J on a patch of area A is:

$$\bar{E}(\bar{r}) = \frac{jk\eta_o}{4\pi} \frac{e^{-jkr}}{r} \int_A \left[(\hat{k} \cdot \bar{J})\, \hat{k} - \bar{J} \right] e^{j\bar{k} \cdot \bar{r}} \, dA$$

where:

$k = 2\pi/\lambda$

$\hat{k} = \bar{r}/|\bar{r}|$

$\bar{k} = k\,\hat{k}$

η_o = 376 (free space impedance) ohms

$\bar{r}$ = vector from patch center to observation point

The total field is found by summing the contribution from each patch and wire segment.

Ground effects are included by means of an image and the appropriate reflection coefficients. The z component of the segment reference direction vector u changes sign for the image as shown in figure 2.

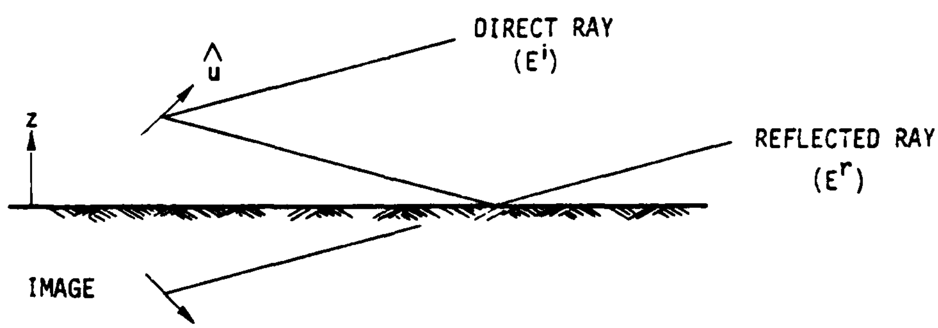


Figure 2. Fields Due to a Segment and Its Image

Using this convention, the reflected electric field can be written in terms of the image field (E^I) as

$$\bar{E}^R = R_\perp (\bar{E}^I \cdot \hat{p}) \hat{p} + R_{||} \left[\bar{E}^I - (\bar{E}^I \cdot \hat{p}) \hat{p} \right]$$

$$= R_{||} \bar{E}^I + (R_\perp - R_{||})(\bar{E}^I \cdot \hat{p}) \hat{p}$$

where $\hat{p}$ is a unit vector perpendicular to the plane of incidence, and

$$R_\perp = \frac{\cos \theta - \sqrt{\varepsilon_E - \sin^2 \theta}}{\cos \theta + \sqrt{\varepsilon_E - \sin^2 \theta}}$$

$$R_{||} = - \frac{\varepsilon_E \cos \theta - \sqrt{\varepsilon_E - \sin^2 \theta}}{\varepsilon_E \cos \theta + \sqrt{\varepsilon_E - \sin^2 \theta}}$$

are the reflection coefficients for the image field perpendicular and parallel, respectively, to the plane of incidence, with

$$\varepsilon_E = \frac{\varepsilon_1}{\varepsilon_0} \left(1 - \frac{j\sigma_1}{\omega\varepsilon_1}\right)$$

and θ measured from $\hat{z}$.

Note: the "electrical field" calculated by the routine does not include the term $\frac{e^{-jkr}}{r}$. That is, in equation (1), $\frac{r}{e^{-jkr}} \bar{E}(\bar{r})$ is actually calculated.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
A	$2 \frac{\sin \pi w_i s}{w_i}$
AII	Location of the imaginary part of A_i
AIR	Location of the real part of A_i

ARG	$j\bar{k} \cdot \bar{R}_i$
B	EL (BOO-TOO)
BII	Location of the imaginary part of B_i
BIR	Location of the real part of B_i
BOO	$(\sin \pi (1 - w_i)s)/(\pi s (1 - w_i))$
BOT	$\pi s (1 - w_i)$
C	EL (BOO + TOO)
CCX, CCY, CCZ	Store CIX, CIY, CIZ
CDP	$(R - R_{\|\|}) (E^I \cdot \hat{p})\hat{p}$
CII	Location of the imaginary part of C_i
CIR	Location of the real part of C_i
CIX, CIY, CIZ	Main use: $\sum_{i=1}^{N} e^{j\bar{k} \cdot \bar{R}_i} \bar{Q}_i$
CONST	$-j (\eta_0/4\pi)$
EL	πs
EPH	$r/(e^{-jkr}) E_\rho$
ETH	$r/(e^{-jkr}) E_\theta$
EXA	Intermediate calculation
INCORE	Logical .TRUE. when A_i, B_i, C_i are stored in core
JX, JY, JZ	X,Y, and z components of current
KDOTJ	$\bar{k} \cdot \bar{J}$
LAI	Location of the imaginary part of A_i
LAR	Location of the real part of A_i
LBI	Location of the imaginary part of B_i

LBR	Location of the real part of B_i
LCI	Location of the imaginary part of C_i
LCR	Location of the real part of C_i
NWIRE	Number of wire segments
OMEGA	w_i
PHI	ϕ in radians
PHX, PHY	X and Y components of $\hat{\phi}$
RI	Imaginary part of Q_i
RR	Real part of Q_i
RRH	$R_{\perp}$
RRV	$R_{\|\|}$
SILL	Intermediate variable
STOR	$j\mu_0\ 2\pi\ c$
THET	θ in radians
THX, THY, THZ	X,Y,Z components of $\hat{\theta}$
TOO	$\frac{\sin \pi(1 + w_i)s}{\pi\ s\ (1 + w_i)}$
TOP	Intermediate variable
ZRSIN	Intermediate variable

5. I/O VARIABLES

A. INPUT	LOCATION
DBGPRT	/ADEBUG/
INCORE	F.P.
IPERF	/AMPZIJ/
KSYMP	/AMPZIJ/

	LOCAII	/FLDCOM/
	LOCAIR	/FLDCOM/
	LOCBII	/FLDCOM/
	LOCBIR	/FLDCOM/
	LOCCII	/FLDCOM/
	LOCCIR	/FLDCOM/
	NUMSEG	/SEGMNT/
	PHI	F.P.
	SEGTBL	/SEGMNT/
	TEMP	/TEMP01/
	THET	F.P.
B.	OUTPUT	LOCATION
	EPH	F.P.
	ETH	F.P.

6. CALLING ROUTINE:

FLDDRV

7. CALLED ROUTINES:

ASSIGN

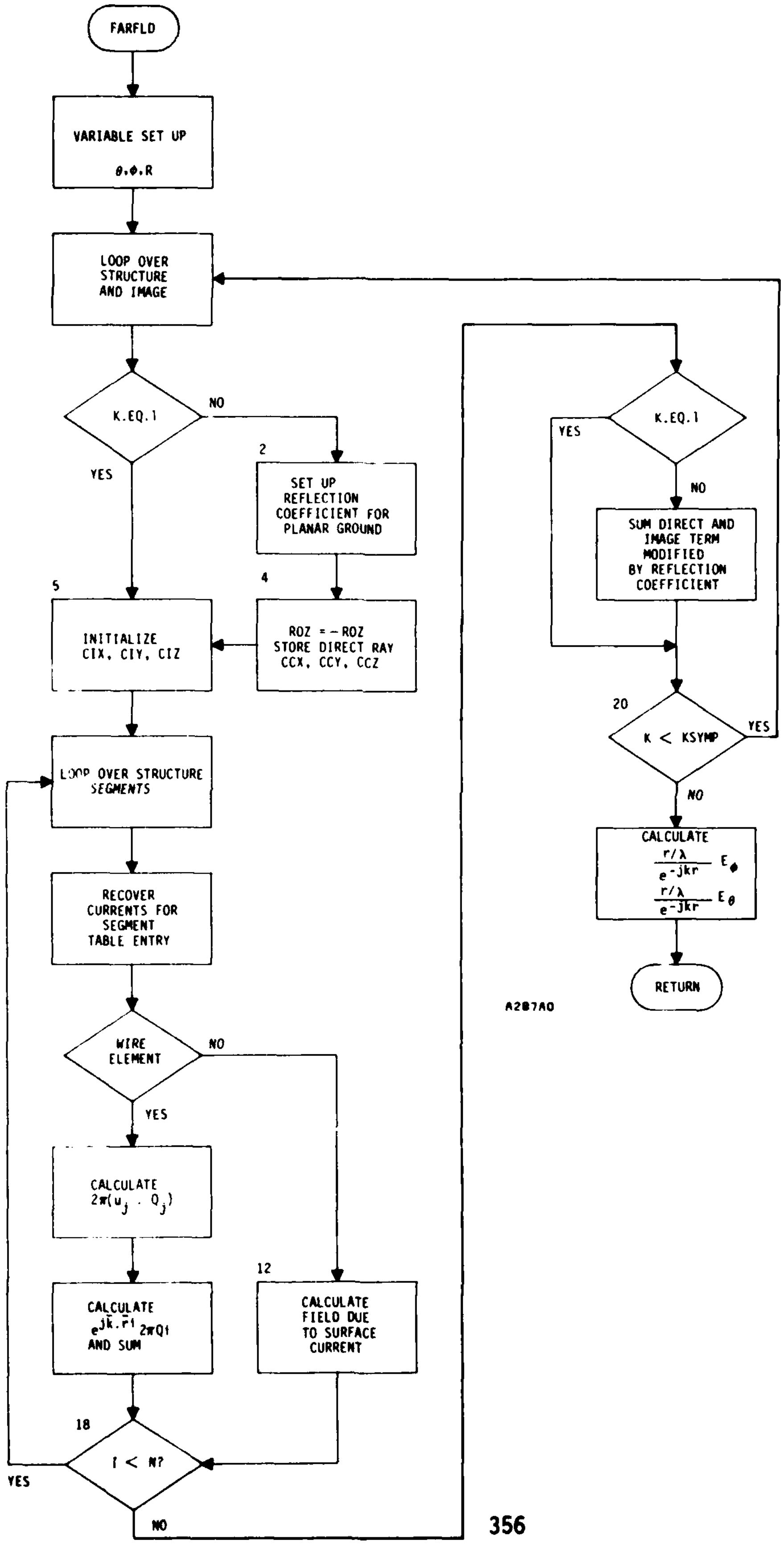
GETSEG

STATIN

STATOT

WLKBCK

FARFLD (MOM)


FARFLD
VARIABLE SET UP
θ,ϕ,R
LOOP OVER STRUCTURE AND IMAGE
K.EQ.1
NO
YES
2
SET UP REFLECTION COEFFICIENT FOR PLANAR GROUND
4
5
INITIALIZE CIX, CIY, CIZ
ROZ = −ROZ STORE DIRECT RAY CCX, CCY, CCZ
LOOP OVER STRUCTURE SEGMENTS
RECOVER CURRENTS FOR SEGMENT TABLE ENTRY
WIRE ELEMENT
NO
YES
CALCULATE 2π(u_j · Q_j)
12
CALCULATE e^{jk·ri} 2πQi AND SUM
CALCULATE FIELD DUE TO SURFACE CURRENT
18
I < N?
YES
NO
K.EQ.1
YES
NO
SUM DIRECT AND IMAGE TERM MODIFIED BY REFLECTION COEFFICIENT
20
K < KSYMP
YES
NO
CALCULATE
r/λ / e^{-jkr} E_ϕ
r/λ / e^{-jkr} E_θ
RETURN
A287A0

1. NAME: FCT (GTD)

2. PURPOSE: This function computes the integrand for various integrals used to compute the diffraction coefficient for an elliptic cylinder.

3. METHOD: Although five different integrands could be calculated, for the present code only the integrand defined for ID equal to three is used. This function calculates FCT which is given by:

$$FCT(x) = A^2 \sin^2 x + B^2 \cos^2 x \ .$$

FCT is used by subroutine DQG32 and by the near-field calculations for determining the creeping wave ray path on the cylinder, to compute the arc length between two points on the elliptic cylinder in a plane. This two-dimensional x-y plane arc length is given by:

$$t = \int_{V_i}^{V_f} FCT(v)dv \ .$$

The driver routines which determine the creeping wave ray path for the cylinder scattered field use the 3-dimensional arc length:

$$t = \frac{1}{|SAS|} \int_{V_i}^{V_f} FCT(v)dv$$

to determine the length of the creeping wave in far-field calculations. SAS is defined in section 4.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
A	Cylinder radius along x-axis
A2	The square of the radius of the elliptic cylinder on the x-axis

B	Cylinder radius along y-axis
B2	The square of the radius of the elliptic cylinder on the y-axis
CS	Cosine of x
F	SQRT((A*SIN(VR))**2+(B*COS(VR))**2)
FCT	Returned integrand
ID	Flag for which FCT function to use
SAS	The sine of π minus THSR, where THSR is the theta angle of the observation direction in RCS relative to the cylinder axis in radians
SN	Sine of x
SNA	The absolute value of the sine of the angle measured from the negative z-axis of the cylinder to the direction of propagation
X	The argument of the integrand defining the elliptic angle

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
A	/GEOMEL/
B	/GEOMEL/
ID	/GTD/
SAS	/GTD/
X	F.P.

B. OUTPUT

OUTPUT	LOCATION
FCT	FUNCTION

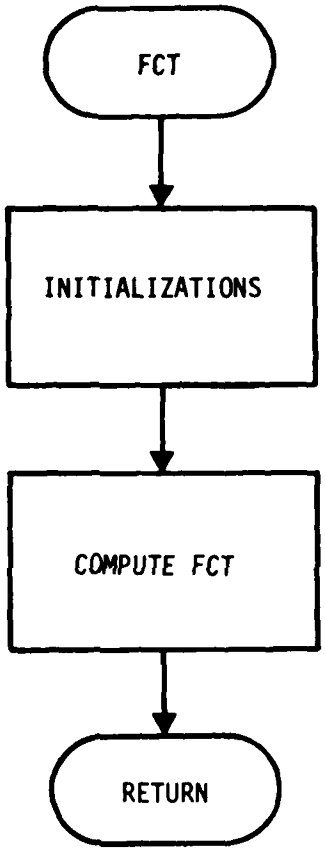
6. CALLING ROUTINES:

DQG32	SCLRPL
RPLSCL	SCTCYL

7. CALLED ROUTINES:

None

FCT (GTD)



1. NAME: FFCT (GTD)

2. PURPOSE: This function is used to calculate the transition function for the corner diffraction coefficient.

3. METHOD: The transition function for the edge and corner diffraction coefficients is given by (see reference A):

$$\mathrm{FFCT}(x) = 2j\left|\sqrt{x}\right| e^{jx} \int_{\left|\sqrt{x}\right|}^{\infty} e^{-j\tau^2}\, d\tau$$

This can also be written as (see reference B):

$$\underbrace{\mathrm{FFCT}(x)}_{\mathrm{FFCT(DEL)}} = j\sqrt{2\pi|x|}\; e^{jx}\left[(0.5-j0.5) - \left(\underbrace{C\underbrace{\sqrt{\frac{2|x|}{\pi}}}_{S}}_{\mathrm{CFR}} - j\underbrace{S\underbrace{\sqrt{\frac{2|x|}{\pi}}}_{S}}_{\mathrm{SFR}}\right)\right]$$

CFR Computed in subroutine FRNELS

SFR Computed in subroutine FRNELS

so that we can use the Fresnel integral:

$$\int_0^{\alpha} e^{-j\frac{\pi}{2}t^2}\, dt = C(\alpha) - jS(\alpha).$$

If the absolute value of x, denoted by DEL, is greater than 10, FFCT is set equal to 1 + 0j.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
CFR	Real part of Fresnel integral
DEL	Argument of transition function

FFCT	Transition function
PI	π
S	Argument of Fresnel integral
SDEL	SQRT (ABS(DEL))
SFR	Imaginary part of Fresnel integral
TPI	2π

5. I/O VARIABLES:

A. INPUT — LOCATION

DEL	F.P.
PI	/PIS/
TPI	/PIS/

B. OUTPUT — LOCATION

FFCT	COMPLEX FUNCTION

6. CALLING ROUTINES:

DICOEF

DIFPLT

DPLRPL

RPLDPL

7. CALLED ROUTINES:

BEXP

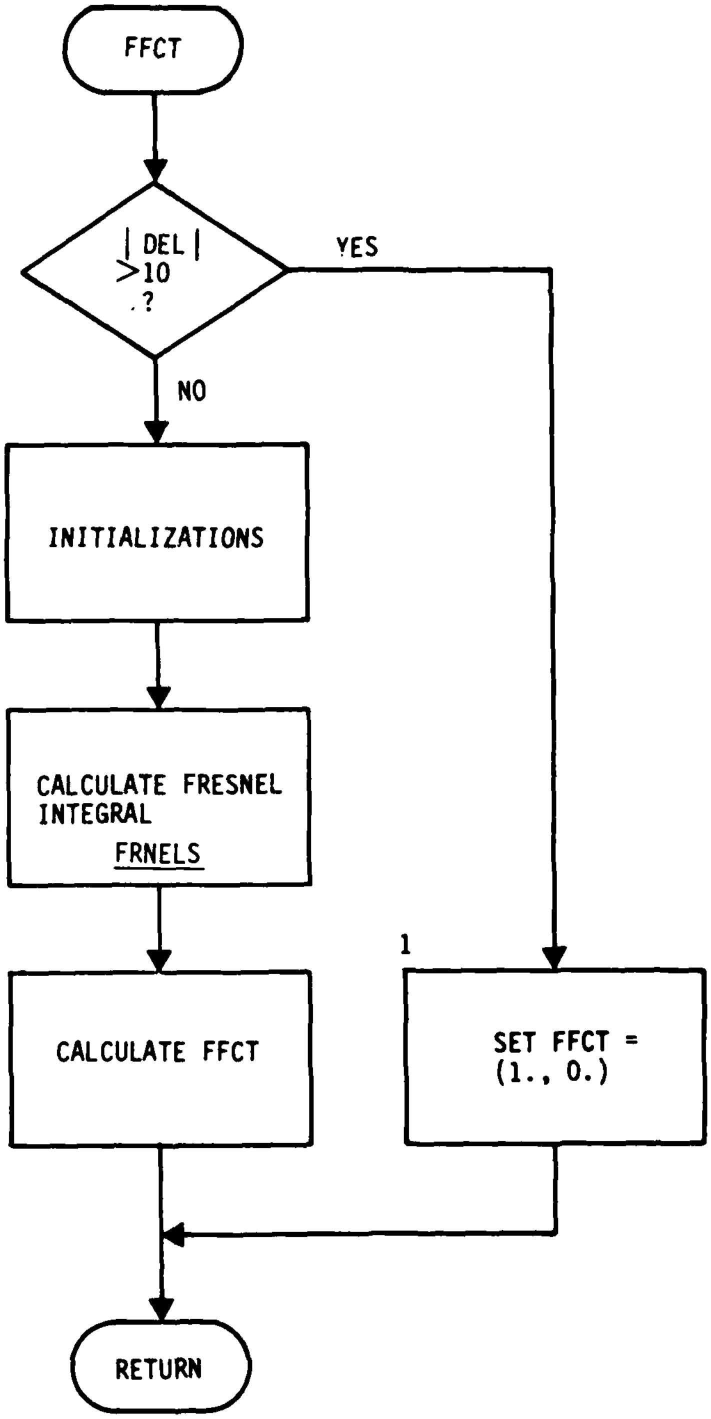
FRNELS

8. REFERENCES:

A. R. G. Kouyoumjian and P. H. Pathak, "A Uniform Geometrical Theory of Diffraction for an Edge in a Perfectly Conducting Surface," Proc. IEEE, Vol. 62, November 1974, pp. 1448-1461.

B. The BDM Corporation, "Interim Report for Addition of GTD to GEMACS," Vol II, pp. E-14 to E-16, Contract F30602-81-C-0084, BDM/A-81-676-TR, January 1982.

FFCT (GTD)


FFCT
| DEL | > 10 ?
YES
NO
INITIALIZATIONS
CALCULATE FRESNEL INTEGRAL
FRNELS
1
CALCULATE FFCT
SET FFCT = (1., 0.)
RETURN

1. NAME: FKARG (GTD)

2. PURPOSE: To compute a parameter needed in the diffraction coefficient for the elliptic cylinder.

3. METHOD: This subroutine computes the parameter used in the diffraction coefficient to determine the fields scattered from the elliptic cylinder. This parameter is given by (see reference A):

$$\xi = \int_{Q_1}^{Q_2} \pi^{1/3} \rho_g^{-2/3} \, dt,$$

where ρ_g is the radius of curvature of the elliptic cylinder in the plane of propagation. This can also be written as:

$$\xi = \pi^{1/3} (AB)^{2/3} |\sin\alpha|^{1/3} \int_{v_i}^{v_f} \frac{dv}{\sqrt{A^2 \sin^2 v + B^2 \cos^2 v}} ,$$

where

ξ = SKWIG

α = ALR

v_i = VIR

v_f = VFR.

FKARG solves the second equation. The constants are computed in FKARG. The iteration is accomplished in subroutine DQG32. The two terms are multiplied in FKARG.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
A	Radius of elliptic cylinder along x-axis
ALR	Angle measured from negative z-axis in the direction of propagation
ANS	The evaluated integral

B	Radius of elliptic cylinder along y-axis
FUNI	Integrand of the integral
PI	π
SKWIG	Parameter used to define curved surface at the point of diffraction
VFR	Elliptical angle defining the diffraction angle position on the cylinder
VIR	Elliptical angle defining the incident angle position on cylinder

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
A	/GEOMEL/
ALR	F.P.
B	/GEOMEL/
PI	/PIS/
VFR	F.P.
VIR	F.P.

B. OUTPUT

OUTPUT	LOCATION
SKWIG	F.P.

6. CALLING ROUTINES:

RPLSCL

SCLRPL

SCTCYL

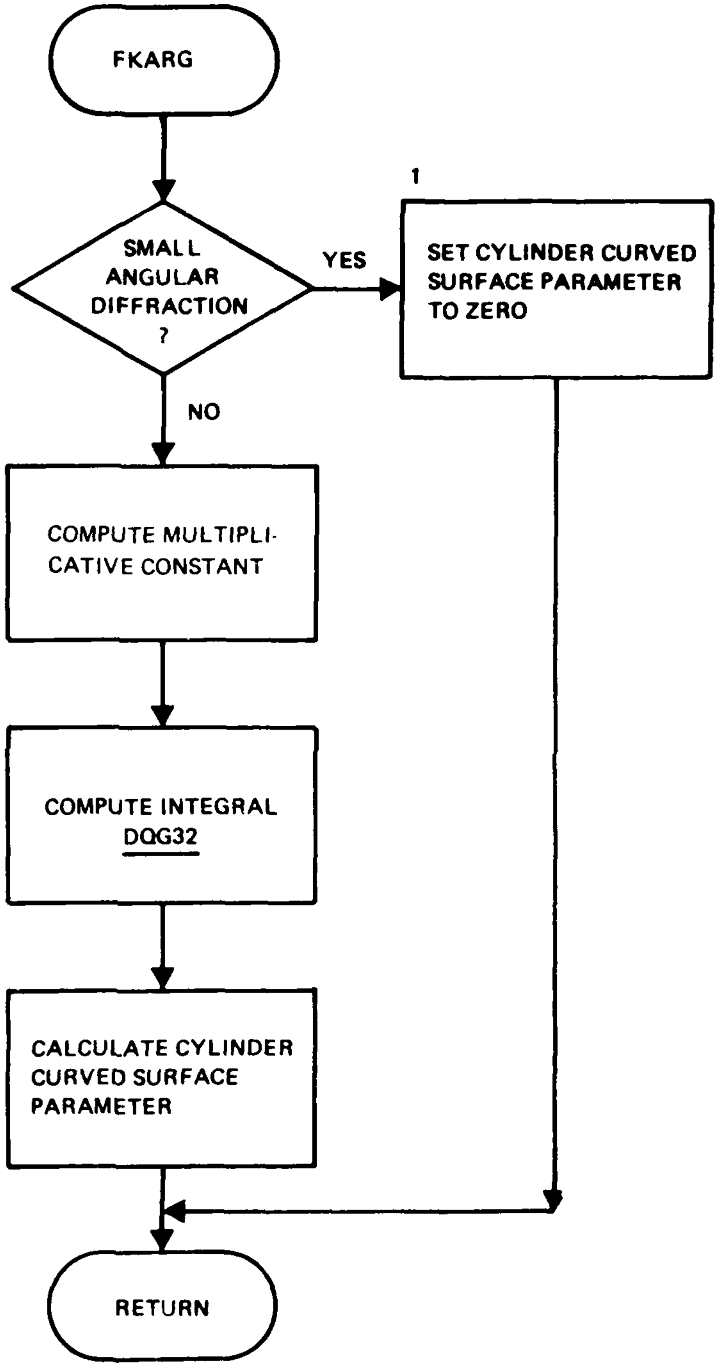
7. CALLED ROUTINES:

DQG32

FUNI

8. REFERENCE:

A. P. H. Pathak, W. D. Burnside, and R. J. Marhefka, "A Uniform GTD Analysis of the Diffraction of Electromagnetic Waves by a Smooth Convex Surface," submitted for publication to <u>IEEE Trans. on Antennas and Propagation</u>. (Also Report 784583-4, April 1979, The Ohio State University ElectroScience Laboratory, Department of Electrical Engineering; prepared under Contract No. N62269-76-C-0554 for Naval Air Development Center.


FKARG
SMALL ANGULAR DIFFRACTION ?
YES
1
SET CYLINDER CURVED SURFACE PARAMETER TO ZERO
NO
COMPUTE MULTIPLI-CATIVE CONSTANT
COMPUTE INTEGRAL DQG32
CALCULATE CYLINDER CURVED SURFACE PARAMETER
RETURN

1. NAME: FKY (GTD)

2. PURPOSE: This function is used in computing the transition function for curved edge diffraction.

3. METHOD: The transition function for the diffraction coefficient of an edge in a curved surface is the same as for a straight wedge, except that the curved edge function takes into account the possibility of the distance parameter being negative. The transition function is given by (see Reference A):

$$F(x) = 2j\left|\sqrt{x}\right|e^{jx} \int_{\left|\sqrt{x}\right|}^{\infty} e^{-j\tau^2}\, d\tau,$$

where $x = kLa$,

and $k = 2\pi/\lambda$

L = distance parameter, which is a function of source type, incident angle, and source-to-diffraction point distance separation; ensures the field will be continuous at the shadow and reflection boundaries

a = a function dependent on the square of the cosine of the incident and diffraction angles and the wedge angle number.

So that the Fresnel integral can be used, the transition function can then be written as (see reference B):

$$\underbrace{F(kLa)}_{\text{FKY(FL,A)}} = j2\pi\sqrt{\frac{|L|a}{\lambda}}\, e^{jk|L|a}\left[(0.5-j0.5) - \left(\underbrace{C\left(\underbrace{2\sqrt{\frac{|L|a}{\lambda}}}_{XS}\right)}_{C} - j\underbrace{S\left(\underbrace{2\sqrt{\frac{|L|a}{\lambda}}}_{XS}\right)}_{S}\right)\right], \text{ for } L>0$$

C: Computed in subroutine FRNELS

S: Computed in subroutine FRNELS

and

$$F(kLa) = F^*(k|L|a), \qquad \text{for } L<0$$

where the "*" means the complex conjugate, and the Fresnel integral is given as:

$$\int_0^{\alpha} e^{-j\frac{\pi}{2}t^2}\,dt = C(\alpha) - jS(\alpha).$$

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
A	Parameter dependent on the incident and diffracted angles
C	Real part of Fresnel integral
FKY	Transition function
FL	The distance parameter in wavelengths
FLA	Absolute value of FL
S	Imaginary part of Fresnel integral
TPI	2π
XS	Argument of Fresnel integral

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
A	F.P.
FL	F.P.
TPI	/PIS/

B. OUTPUT

OUTPUT	LOCATION
FKY	FUNCTION

6. CALLLING ROUTINE:

   DZCOEF

7. CALLED ROUTINES:

   BEXP

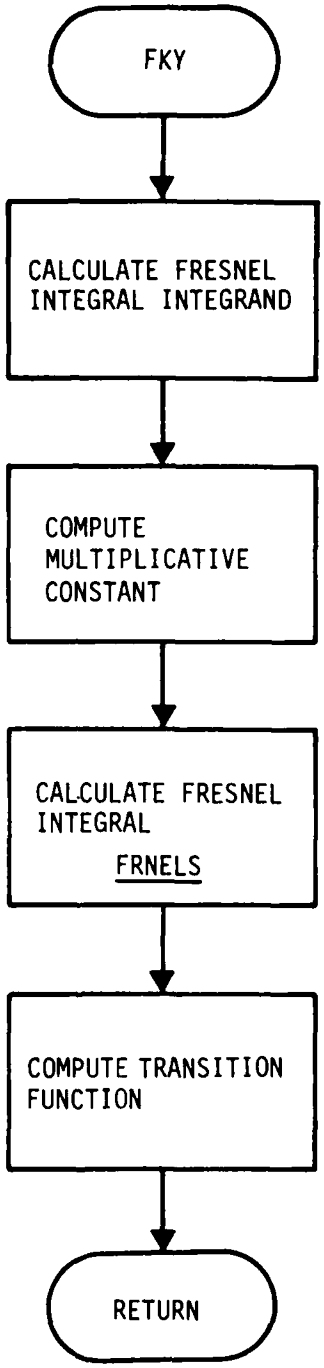
   FRNELS

8. REFERENCES:

A. R. G. Kouyoumjian and P. H. Pathak, "A Uniform Geometrical Theory of Diffraction for an Edge in a Perfectly Conducting Surface," Proc. IEEE, Vol. 62, November 1974, pp. 1448-1461.

B. The BDM Corporation, "Interim Report for Addition of GTD to GEMACS," Vol. II, pp. E-14 to E-16, Contract F30602-81-C-0084, BDM/A-81-676-TR, January 1982.

FKY (GTD)


FKY
CALCULATE FRESNEL INTEGRAL INTEGRAND
COMPUTE MULTIPLICATIVE CONSTANT
CALCULATE FRESNEL INTEGRAL
FRNELS
COMPUTE TRANSITION FUNCTION
RETURN

1. NAME: FLDDRV (GTD)

2. PURPOSE: To calculate the electric field scattered by a structure or incident from a source at points specified by the user.

3. METHOD: Two data sets are calculated by this subroutine. If any GTD interactions have been requested, FLDDRV first calculates the Green's function matrix of field values due to unit currents on the MOM structure. These fields take into account the presence of the GTD portion of the geometry. If only MOM or GTD objects are in the geometry, the matrix is not computed.

   The second data set that may be calculated by FLDDRV is the incident field matrix of field values due to all sources which contributed to the solution current specified in the EFIELD command. This data set is generated only when an incident field interaction (EI, ES, OR EU) has been requested on a previous SETINT command. Otherwise, the field matrix is set equal to zero.

   FLDDRV will accept coordinates for three different systems: Cartesian, cylindrical, and spherical. The initial coordinates will be received in a specific order. This order determines the order in which each coordinate is incremented. The following is an example command:

   NEAR = EFIELD (CURDEN) LINLIN

DX = 1.	DY = 10.	DZ = 10.
X2 = 10.	Y2 = 10.	Z2 = 10.
Z1 = 0.	Y1 = 0.	X1 = 0.

   Z will be on the outer loop. Y will be on the middle loop, and X will be on the inner loop. This is determined by looking at the initial coordinates Z1, Y1, and X1. The total number of field points in this case is 11 x 2 x 2 = 44. The number of field values that must be calculated is 132 (44 x 3 polarizations).

   The formats of the Green's function and incident field matrices are shown in figure 1. These matrices will be used by FLDDRV (MOM) to calculate the total radiated field:

$$\underline{F}^t = \underline{\underline{G}}^T \underline{I} = \underline{F}^i$$

$\underline{I}$	=	Solution Vector
$\underline{F}^t$	=	Total Field
$\underline{F}^i$	=	Incident Field
$\underline{\underline{G}}^T$	=	Green's Function Matrix, Transposed.

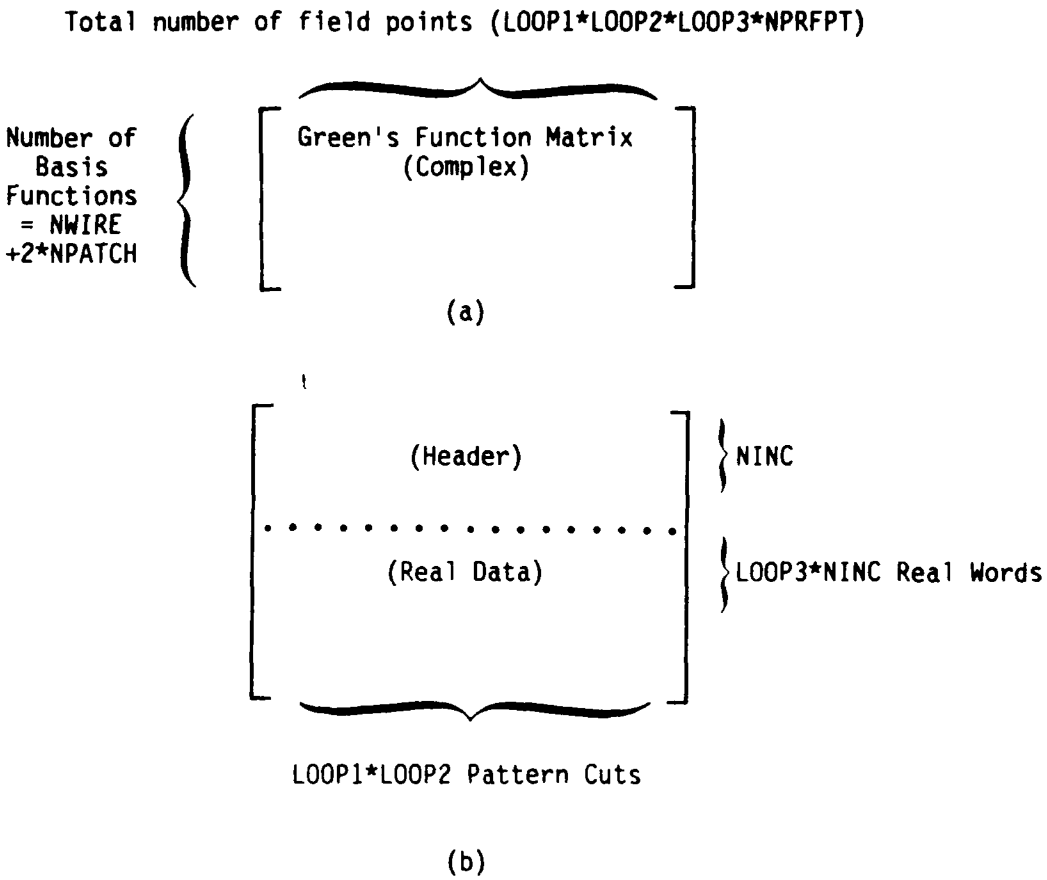


Figure 1. The Formats of (a) Green's Function Matrix and (b) Incident Field Matrix.

MOM-only problems assume $\underline{F}^i = 0$. GTD-only problems assume $\underline{\underline{G}} = 0$ and $\underline{I} = 0$.

The data set formats have been made compatible with the MOM and OUTPUT modules so that the GTD module will interface directly with the OUTPUT module for GTD-only problems.

FLDDRV may be used with either a solution or source data set as the second item on an EFIELD command. A solution data set generates the Green's function matrix. The solution data set lineage is searched for a geometry data set. Unit currents are placed on each MOM segment and the radiated field calculated. Each element of $\underline{\underline{G}}$ is computed from:

$$g_{ij} = \frac{E_j}{I_i} \Bigg| \text{ all other } I = 0$$

E_j is a tangential component of the electric field at one of the points specified on the EFIELD command. I_i is a unit current impressed on the ith MOM segment. The actual calculation of g_{ij} is performed by ZGTDRV. The current on all other MOM segments is set equal to zero.

A solution data set may also trigger generation of the incident field matrix, as does the source data set. The lineage of the solution data set is searched for a source data set. If one is found (and incident fields have been requested), an incident field matrix is generated by computing the field at points specified on the EFIELD command due to the source which generated the source data set. This is also done by ZGTDRV. A header is placed at the beginning of each field record (pattern cut) for use by the OUTPUT module in printing and plotting the field patterns.

4. INTERNAL VARIABLES:

VARIABLE	DEFINITION
CC2	C2 in degrees
CC3	C3 in degrees
C1	First coordinate of field point (R or X)
C2	Second coordinate of field point (θ or Y)
C3	Third coordinate of field point (ϕ or Z)
D	A dummy variable used in the call to ZGTDRV
DC	Step size for inner loop (pattern cut increment)
FRFLD	Logical far-field pattern flag
I	Loop index
I1	First column of Green's function matrix in TEMP

IBITB	Attribute word of the solution or source data set (NAMEB)
IBLANK	Hollerith field with all blank characters
IBT	Flag for a solution or source data set
ICOL2	Last column of Green's function matrix in TEMP
ICORDT	Coordinate keyword table used to find all the coordinate positions and increments
ICOST	Coordinate order and system table. This table will tell which coordinates are required for a coordinate system (Cartesian, cylindrical, or spherical). Should an improper coordinate type be specified, an error will be generated. For near-field patterns, it also determines the order of coordinates.
ICTYPE	Type of coordinate system, formed by adding the location of system constitutive parameters in ICOST: 6 = rectangular (1+2+3) 12 = cylindrical (4+5+3) 15 = spherical (4+5+6)
IGEOBT	Flag for geometry data link
IGFM	Rightmost three characters of the Green's function data set = "GFM"
II	Number of GTD and MOM interactions in KJ interaction array
INCORE	Logical flag which indicates that interpolation coefficients are stored in main memory
INDEX	Saves the order of coordinates as specified by the user
INDEX1	Index to the U array for the first coordinate
INDEX2	Index to the U array for the second coordinate

INDEX3	Index to the U array for the third coordinate
INDX	Index to the symbol table
INDXA	Index to the field data set symbol table entry
INDXB	Index to the solution or source data set symbol table entry
IOBS1	First observation point number for call to ZGTDRV
IOBS2	Last observation point number for call to ZGTDRV
IROWA	Number of rows in a column of the field data set
ISRCBT	Flag indicating if a data set is a source data set
ITYPE	GTD interaction type. For scattered fields, ITYPE = 3 and the Green's function matrix is generated. For incident fields, ITYPE = 4 and the incident field matrix is generated
IU	The coordinate information array (equivalenced to U)
J	Loop index
JSAV	Index to ICORDT for coordinate system type
JSRC1	First source point for call to ZGTDRV
JSRC2	Last source point for call to ZGTDRV
K	Loop index
KCOLS	Number of columns of the Green's function matrix which will fit into TEMP at once
KJ	The interaction array passed to ZGTDRV for ITYPE = 3 or ITYPE = 4
KWA	Keyword argument

L	Loop index
LES	Logical flag indicating shadowed incident fields are to be included in the incident field calculation (ITYPE = 4)
LEU	Logical flag indicating unshadowed incident fields are to be included in the incident field calculation (ITYPE = 4)
LINKA	Index of field matrix to link the field matrix to the solution or source data set
LINKB	Index to the data set that is linked to the solution or source data set
LINKG	Index to the data set that is linked to the Green's function matrix
LMID	Index of middle pattern loop
LNKBIT	The attribute word for the linked data set
LOOP	The loop array containing the number of times to perform each loop
LOOP1	Outer loop limit
LOOP2	Middle loop limit
LOOP3	Inner loop limit
LORDER	Array containing the order for all coordinate systems
LOUT	Index of outer pattern loop
LX	Total number of pattern cuts stored in the field data set
L123	Total number of field points (LOOP1*LOOP2*LOOP3)
MASK	Used to determine the required coordinates for a system
N	Loop index
NA	Hollerith format of field data set

NAM	Hollerith format of NAMEA
NAMEA	Name of the field data set
NAMEB	Name of the solution or source data set
NAMGEO	Name of the linking geometry data set
NAMGFM	Name of the Green's function matrix
NBITA	Far- or near-field attribute for the field matrix
NC	Number of columns in TEMP for call to ZGTDRV
NCOLS	Total number of columns in the Green's function data set
NDX	Pointer to coordinate number of INDEX array
NDXINR	Index to the coordinate in the U array and in the inner loop
NDXMID	Index to the coordinate in the U array and in the middle loop
NDXOUT	Index to the coordinate in the U array and in the outer loop
NEED	Minimum main memory storage needed to generate a column of the Green's function matrix or field matrix
NEL	TEMP element being initialized to zero
NELRW	Number of elements per row of the data set
NELTTL	Total number of elements of data set in TEMP
NINC	The increment size for filling a column in the field matrix (Number of real words per field point)
NOBS	Number of observation points for the entire Green's function matrix (=LOOP1*LOOP2*LOOP3)
NPRFPT	Number of vector components of field at observation point

NROWS	Number of rows of Green's function matrix
NROWX	Internal variable equal to NROWS for call to ZGTDRV
NS	Hollerith format of geometry data set name (NAMGEO)
NSHIFT	Mask used to blank out the rightmost three characters of the field data set in order to generate the Green's function matrix name
NXTARG	Next argument to be evaluated in the argument list
RDTODG	Conversion factor from radians to degrees
U	Array containing initial and final positions plus the increment for each coordinate

5. I/O VARIABLES:

A. INPUT	LOCATION
CHKWRT	/SYSFIL/
DGTORD	/GEODAT/
FLTARG	/ARGCOM/
INTARG	/ARGCOM/
IPASS	/ARGCOM/
IP217	/GEODAT/
ISON	/ADEBUG/
KBCPLX	/PARTAB/
KBFFLD	/PARTAB/
KBGEOM	/PARTAB/
KBNFLD	/PARTAB/

FLDDRV (GTD)

KBREAL	/PARTAB/
KBSOLN	/PARTAB/
KBSRCE	/PARTAB/
KJFLD	/INTMAT/
KJGTD	/INTMAT/
KJINT	/INTMAT/
KJMOM	/INTMAT/
KOLBIT	/PARTAB/
KOLCOL	/PARTAB/
KOLLNK	/PARTAB/
KOLNAM	/PARTAB/
KWNAME	/PARTAB/
LSTSYS	/SYSFIL/
LUPRNT	/ADEBUG/
NBYTSZ	/ADEBUG/
NCODES	/PARTAB/
NDATBL	/PARTAB/
NOPCOD	/ADEBUG/
NPATCH	/SEGMNT/
NPDATA	/PARTAB/
NTEMPS	/TEMP01/
NTFLPT	/ADEBUG/
NUMARG	/ARGCOM/
NUMGTD	/GTDDAT/
NWIRE	/SEGMNT/

RSTART	/SYSFIL/
ZERO	/ADEBUG/
B. OUTPUT	LOCATION
CHKWRT	/SYSFIL/
IERRF	/ADEBUG/
INTARG	/ARGCOM/
ITEMP	/TEMP01/
IWORDS	/ADEBUG/
LSTSYS	/SYSFIL/
L1	/FLDVAL/
L2	/FLDVAL/
L3	/FLDVAL/
NAMSRC	/FLDVAL/
NDATBL	/PARTAB/
NOGOFG	/SCNPAR/
NPDATA	/PARTAB/
NU	/FLDVAL/
TEMP	/TEMP01/
U1	/FLDVAL/
V1	/FLDVAL/
W1	/FLDVAL/

6. CALLING ROUTINE:

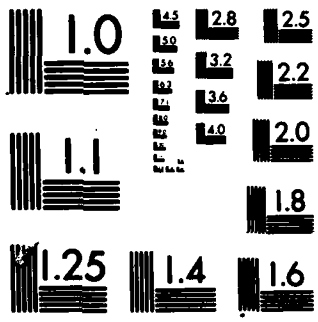
TSKXQT

AD-A137 463 GENERAL ELECTROMAGNETIC MODEL FOR THE ANALYSIS OF COMPLEX SYSTEMS (GEMACS..(U) BDM CORP ALBUQUERQUE NM D L KADLEC ET AL. SEP 83 BDM/A-83-020-TR

5/5

UNCLASSIFIED RADC-TR-83-217-VOL-3 F30602-81-C-0084 F/G 20/14 NL

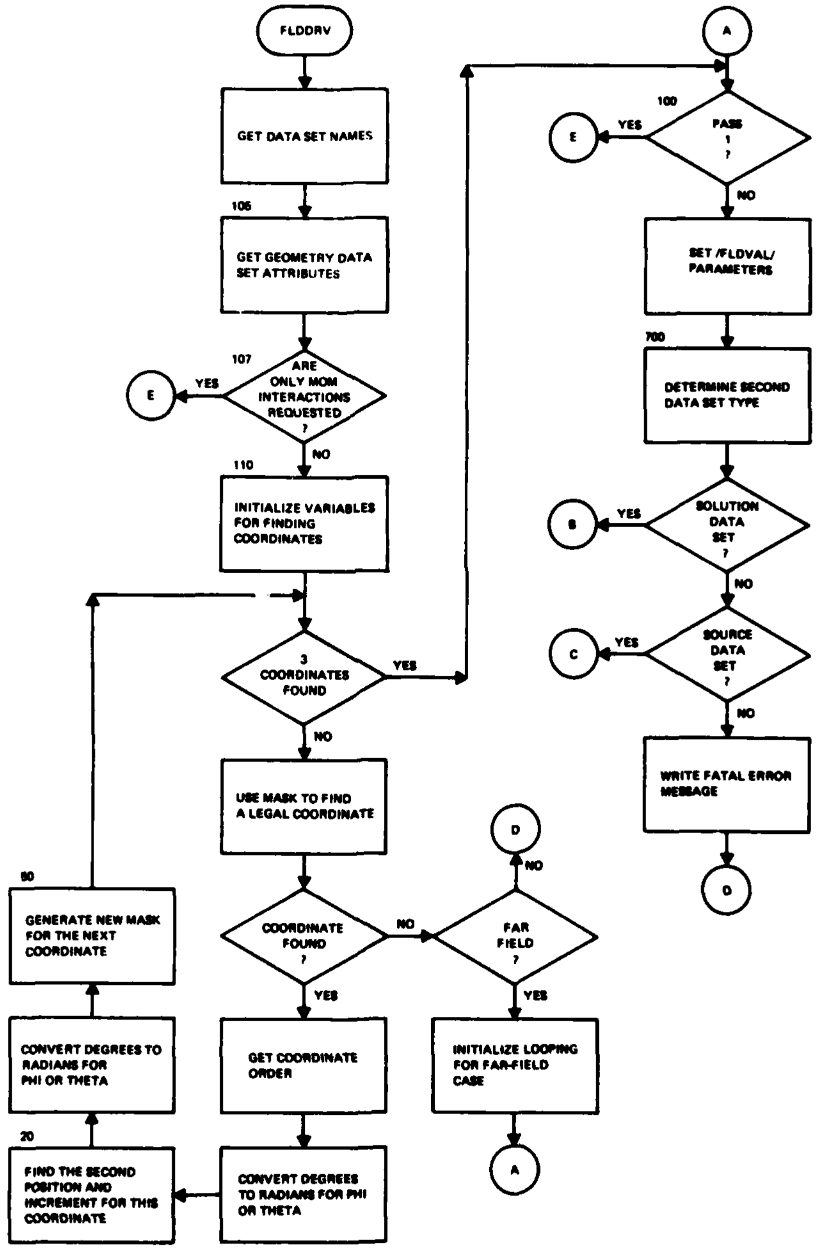
END
DATE
FILMED
3-84
DTIC

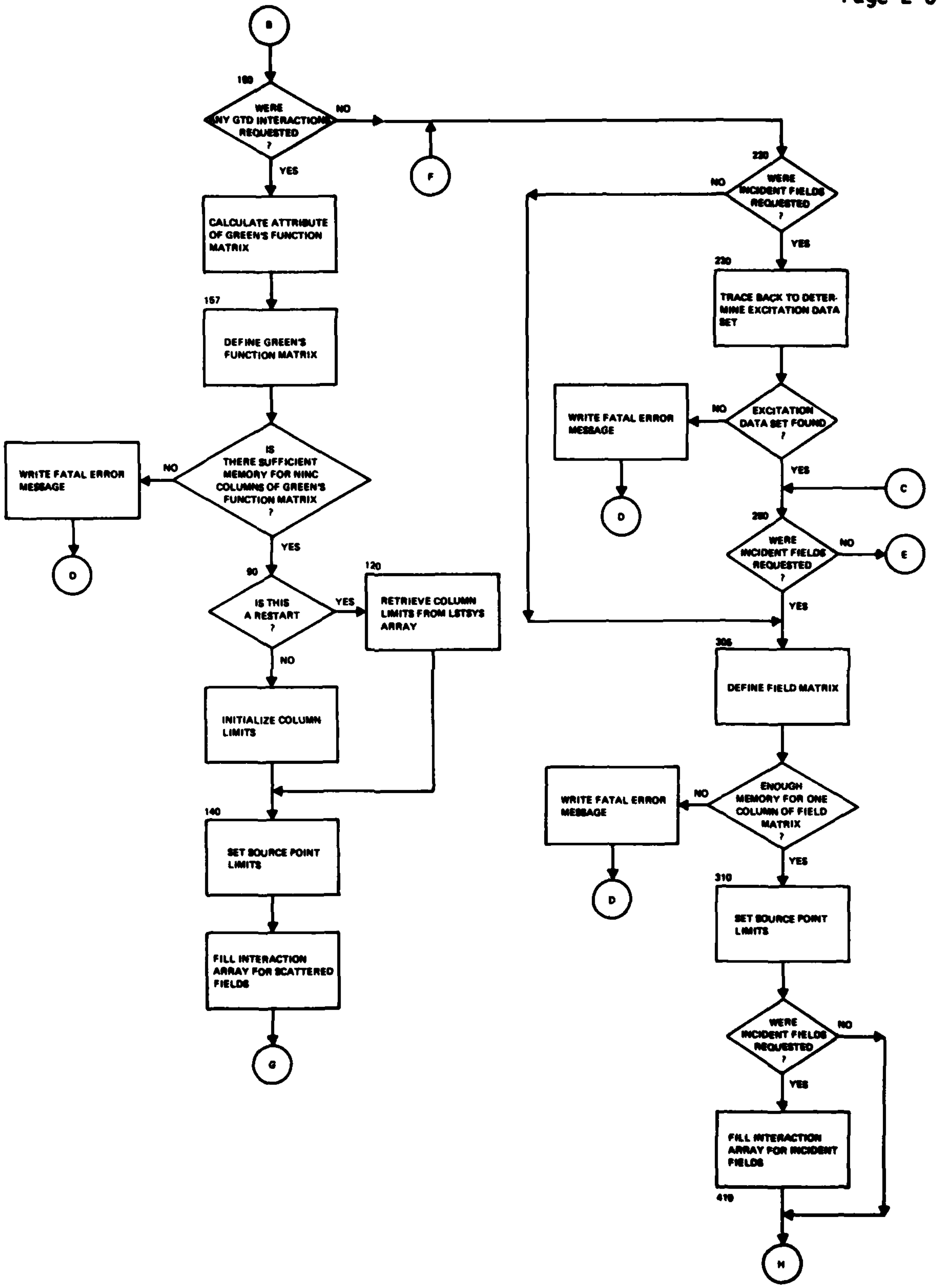


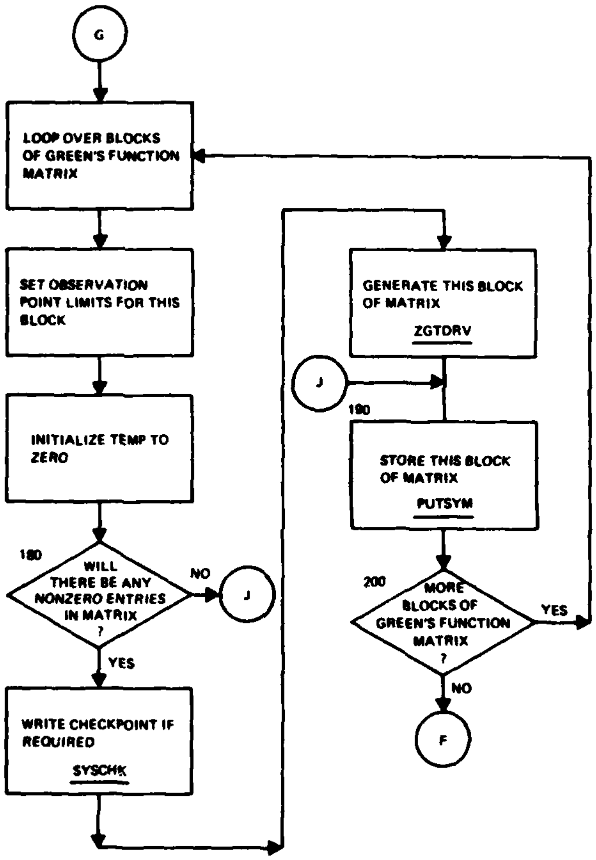
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

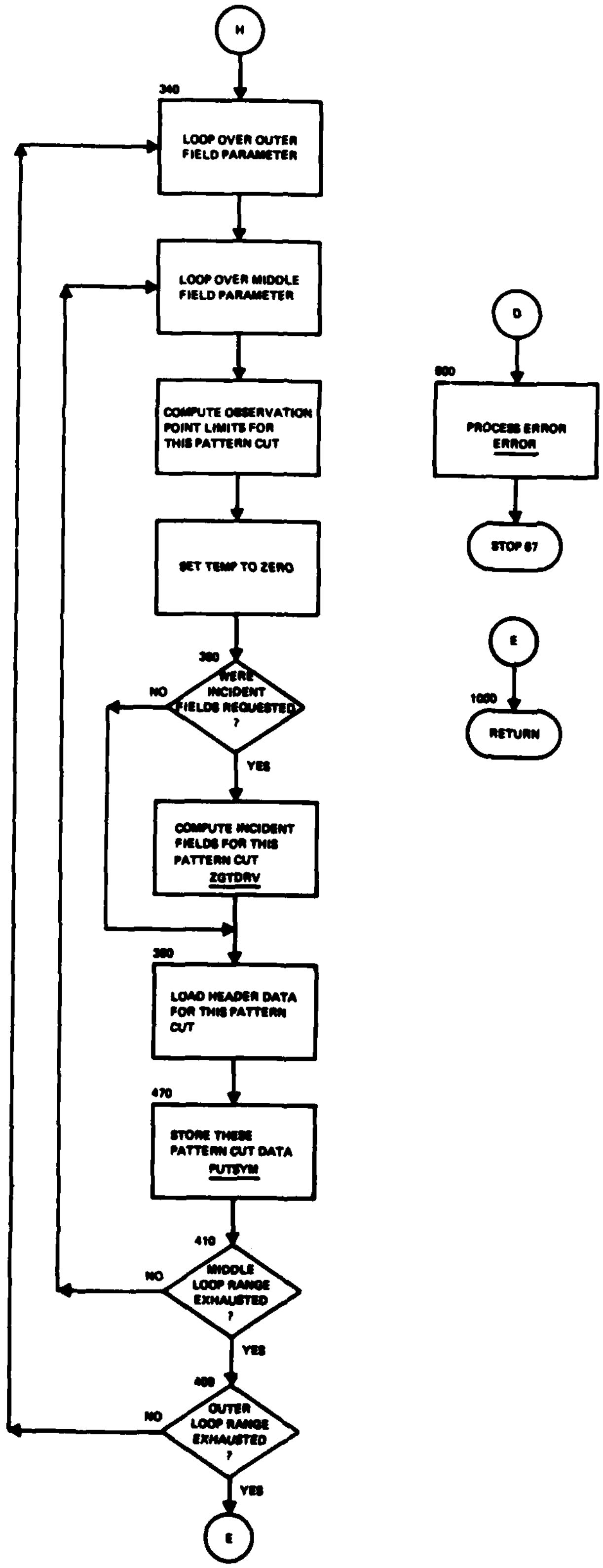
7. CALLED ROUTINES:

ASSIGN	IBITCK	SYMDEF
CONVRT	PRTKJ	SYMUPD
ERROR	PUTSYM	SYSCHK
GETARG	STATIN	WLKBCK
GETGEO	STATOT	ZGTDRV


FLDDRV
GET DATA SET NAMES
105
GET GEOMETRY DATA SET ATTRIBUTES
107
ARE ONLY MOM INTERACTIONS REQUESTED ?
YES
E
NO
110
INITIALIZE VARIABLES FOR FINDING COORDINATES
3 COORDINATES FOUND
YES
NO
USE MASK TO FIND A LEGAL COORDINATE
COORDINATE FOUND ?
NO
FAR FIELD ?
NO
D
YES
INITIALIZE LOOPING FOR FAR-FIELD CASE
A
YES
GET COORDINATE ORDER
CONVERT DEGREES TO RADIANS FOR PHI OR THETA
20
FIND THE SECOND POSITION AND INCREMENT FOR THIS COORDINATE
CONVERT DEGREES TO RADIANS FOR PHI OR THETA
80
GENERATE NEW MASK FOR THE NEXT COORDINATE
A
100
PASS 1 ?
YES
E
NO
SET /FLDVAL/ PARAMETERS
700
DETERMINE SECOND DATA SET TYPE
SOLUTION DATA SET ?
YES
B
NO
SOURCE DATA SET ?
YES
C
NO
WRITE FATAL ERROR MESSAGE
D


B
WERE ANY GTD INTERACTIONS REQUESTED ?
NO
YES
F
CALCULATE ATTRIBUTE OF GREEN'S FUNCTION MATRIX
157
DEFINE GREEN'S FUNCTION MATRIX
IS THERE SUFFICIENT MEMORY FOR NINC COLUMNS OF GREEN'S FUNCTION MATRIX ?
WRITE FATAL ERROR MESSAGE
D
90
IS THIS A RESTART ?
120
RETRIEVE COLUMN LIMITS FROM LSTSYS ARRAY
INITIALIZE COLUMN LIMITS
140
SET SOURCE POINT LIMITS
FILL INTERACTION ARRAY FOR SCATTERED FIELDS
G
220
WERE INCIDENT FIELDS REQUESTED ?
230
TRACE BACK TO DETERMINE EXCITATION DATA SET
EXCITATION DATA SET FOUND ?
WRITE FATAL ERROR MESSAGE
C
WERE INCIDENT FIELDS REQUESTED ?
E
DEFINE FIELD MATRIX
ENOUGH MEMORY FOR ONE COLUMN OF FIELD MATRIX ?
WRITE FATAL ERROR MESSAGE
310
SET SOURCE POINT LIMITS
WERE INCIDENT FIELDS REQUESTED ?
FILL INTERACTION ARRAY FOR INCIDENT FIELDS
H


G
LOOP OVER BLOCKS OF GREEN'S FUNCTION MATRIX
SET OBSERVATION POINT LIMITS FOR THIS BLOCK
INITIALIZE TEMP TO ZERO
180
WILL THERE BE ANY NONZERO ENTRIES IN MATRIX ?
NO
J
YES
WRITE CHECKPOINT IF REQUIRED
SYSCHK
GENERATE THIS BLOCK OF MATRIX
ZGTDRV
J
190
STORE THIS BLOCK OF MATRIX
PUTSYM
200
MORE BLOCKS OF GREEN'S FUNCTION MATRIX ?
YES
NO
F


H
340
LOOP OVER OUTER FIELD PARAMETER
LOOP OVER MIDDLE FIELD PARAMETER
COMPUTE OBSERVATION POINT LIMITS FOR THIS PATTERN CUT
SET TEMP TO ZERO
WERE INCIDENT FIELDS REQUESTED ?
NO
YES
COMPUTE INCIDENT FIELDS FOR THIS PATTERN CUT
ZGTDRV
LOAD HEADER DATA FOR THIS PATTERN CUT
470
STORE THESE PATTERN CUT DATA
PUTSYM
410
MIDDLE LOOP RANGE EXHAUSTED ?
NO
YES
OUTER LOOP RANGE EXHAUSTED ?
NO
YES
E
D
PROCESS ERROR
ERROR
STOP 87
E
1000
RETURN

1. NAME: FLDDRV (MOM)

2. PURPOSE: To calculate the electric field scattered by a structure or incident from a source at points prescribed by the user.

3. METHOD: FLDDRV will accept coordinates for three different systems: Cartesian, cylindrical, and spherical. The initial coordinates will be received in a specific order. This order determines the order in which each coordinate is incremented. The following is an example command:

```
NEAR = EFIELD (CURDEN) LINLIN

          DX = 1.   DY = 10.   DZ = 10.
          X2 = 10.  Y2 = 10.   Z2 = 10.
          Z1 = 0.   Y1 = 0.    X1 = 0.
```

Z will be on the outermost loop. Y will be on the middle loop, and X will be on the innermost loop. This is determined by looking at the initial coordinates Z1, Y1, and X1.

For each position the electric field is calculated for near or far field. The electric field vector components (real and imaginary) are put into a matrix by columns each time the innermost loop is completed. This is equivalent to one column of data per field pattern cut. A header is put on each column to indicate the starting point and which coordinate is being incremented. A matrix of the electric field components for all positions specified will be generated.

For MOM-only interactions, the field values are generated by NERFLD and FARFLD. If incident fields or GTD interactions have been requested, FLDDRV calls EGFMAT to compute the total field from the Green's function matrix and/or incident field matrix previously computed in the GTD module.

4. INTERNAL VARIABLES:

VARIABLES	DEFINITION
CC2	C2 in degrees
CC3	C3 in degrees
CEX	Imaginary part of EX
CEY	Imaginary part of EY
CEZ	Imaginary part of EZ

CI	Array containing imaginary part of the electric field for a coordinate
COEF	Coefficient used in converting Cartesian to spherical electric components
COSC2	Cosine of C2
COSC3	Cosine of C3
CR	Array containing real part of the electric field for a coordinate
C1	First coordinate for one of the systems (X or R)
C2	Second coordinate (Y or theta)
C3	Third coordinate (Z or phi)
DC	Step size for innermost loop
EPH	Complex electric field of far-field phi vector component
ESQR	Magnitude of the electric field squared
ETH	Complex electric field of far-field theta vector component
EX	Complex electric field of first near-field vector component
EY	Complex electric field of second near-field vector component
EZ	Complex electric field of third near-field vector component
FRFLD	Logical far-field flag
I	Loop index
IBLANK	Hollerith field with all blank characters
ICOL1	ICOLMN plus 1
ICOLA	Number of columns added to the field matrix

ICOLMN	Column counter for the output field matrix
ICORDT	Coordinate keyword table used to find all the coordinate positions and increments
ICOST	Coordinate order and system table. This table will tell which coordinates are required for a system (Cartesian, cylindrical, and spherical). Should an improper coordinate type be specified, an error is generated. It also will get the order for near-field patterns.
ICTYPE	Type of coordinate system, formed by adding the location of system constitutive parameters in ICOST 6 = rectangular (1 + 2 + 3) 12 = cylindrical (4 + 5 + 3) 15 = spherical (4 + 5 + 6)
IFILE	Data file on which field data set resides
IGEOBT	Flag for geometry data link
INCORE	Logical flag which indicates that interpolation coefficients are stored in main memory
INDEX	Saves the order of the coordinates
INDEX1	Index to the U array for the first coordinate
INDEX2	Index to the U array for the second coordinate
INDEX3	Index to the U array for the third coordinate
INDX	Index to the symbol table
INDXA	Index to the field data set in symbol table
INDXB	Index to the solution or source data set in symbol table
IPLOT	Plot type to be used by the plotting routine

IROWA	Number of rows in a column
IROW2	Number of rows in the solution matrix
IU	The coordinate information array (equivalenced to U)
J	Loop index
JSAV	Index to ICORDT for coordinate system type
K	Loop index
KWA	Keyword argument
LINKA	Index of the field matrix to link the field matrix to the solution data
LINKB	Index to the data that is linked to the solution data
LNKBIT	The attribute word for the linked data set
LOCAII	Location of imaginary part of A
LOCAIR	Location of real part of A
LOCBII	Location of imaginary part of B
LOCBIR	Location of real part of B
LOC1	The last location of the ABC arrays
LOCCII	Location of imaginary part of C
LOCCIR	Location of real part of C
LOOP	The loop array containing the number of times to perform each loop
LOOP1	Outermost loop limit
LOOP2	Middle loop limit
LOOP3	Innermost loop limit
LOPINR	Index of innermost pattern loop
LOPMID	Index of middle pattern loop

LOPOUT	Index of outermost pattern loop
LORDER	Array containing the order for all coordinate systems
MASK	Used to determine the required coordinates for a system
N	Loop index
NAM	Hollerith format of NAMEA
NAME	Name of the solution data set
NAMEA	Name of the field data set
NAMEB	Name of the solution or source data set
NAMEG	Name of the linking geometry data
NBITA	Far- or near-field attribute for the plot matrix
NDX	Pointer to coordinate number in INDEX array
NDXINR	Index to the coordinate in the U array and in the innermost loop
NDXMID	Index to the coordinate in the U array and in the middle loop
NDXOUT	Index to the coordinate in the U array and in the outermost loop
NINC	The increment size for filling a column in the field matrix (Number of real words per field point)
NPI	Index for imaginary part
NPIC	Index for imaginary part of the field
NPIR	Index for real part of the field
NPRFPT	NINC/2 (Number of vector components at field point)
NXTARG	Next argument to be evaluated

PHSMAG	Phase and magnitude component parts of the complex electric field for each position
PWRMAX	Maximum power of the structure
RDTODG	Conversion factor from radians to degrees
REX	Real part of EX
REY	Real part of EY
REZ	Real part of EZ
RINV	Inverse of projections of unit radius onto x-y plane
SINC2	Sine of C2
SINC3	Sine of C3
U	Array containing initial and final positions plus the increment for each coordinate
U1X, U1Y, U1Z	X,Y, and Z components of first near-field polarization unit vector
U2X, U2Y, U2Z,	X,Y, and Z components of second near- field polarization unit vector
U3X, U3Y, U3Z,	X,Y, and Z components of third near-field polarization unit vector
X	The x coordinate
XW	The x coordinate scaled to wavelength
Y	The y coordinate
YW	The y coordinate scaled to wavelength
Z	The z coordinate
ZW	The z coordinate scaled to wavelength

5. I/O VARIABLES:

A. INPUT

INPUT	LOCATION
DGTORD	/GEODAT/
FLTARG	/ARGCOM/
INTARG	/ARGCOM/
IPASS	/ARGCOM/
ISON	/ADEBUG/
KBFFLD	/PARTAB/
KBGEOM	/PARTAB/
KBNFLD	/PARTAB/
KBREAL	/PARTAB/
KJFLD	/INTMAT/
KJGTD	/INTMAT/
KOLBIT	/PARTAB/
KOLLNK	/PARTAB/
KOLLOC	/PARTAB/
KOLNAM	/PARTAB/
KOLROW	/PARTAB/
KWNAME	/PARTAB/
LUPRNT	/ADEBUG/
NCODES	/PARTAB/
NDATBL	/PARTAB/
NOPCOD	/ADEBUG/
NPDATA	/PARTAB/
NTEMPS	/TEMP01/
NTFLPT	/ADEBUG/

	NUMARG	/ARGCOM/
	ZERO	/ADEBUG/
B.	OUTPUT	LOCATION
	IERRF	/ADEBUG/
	ITEMP	/TEMP01/
	IWORDS	/ADEBUG/
	LOCAII	/FLDCOM/
	LOCAIR	/FLDCOM/
	LOCBII	/FLDCOM/
	LOCBIR	/FLDCOM/
	LOCCII	/FLDCOM/
	LOCCIR	/FLDCOM/
	NDATBL	/PARTAB/
	NOGOFG	/SCNPAR/
	NPDATA	/PARTAB/
	TEMP	/TEMP01/

6. CALLING ROUTINE:

TSKXQT

7. CALLED ROUTINES:

ASSIGN

CABC

CLSFIL

CONVRT

EGFMAT

ERROR

FARFLD

FLDDRV (MOM)

GETARG

GETGEO

GETSYM

IBITCK

NERFLD

PUTSYM

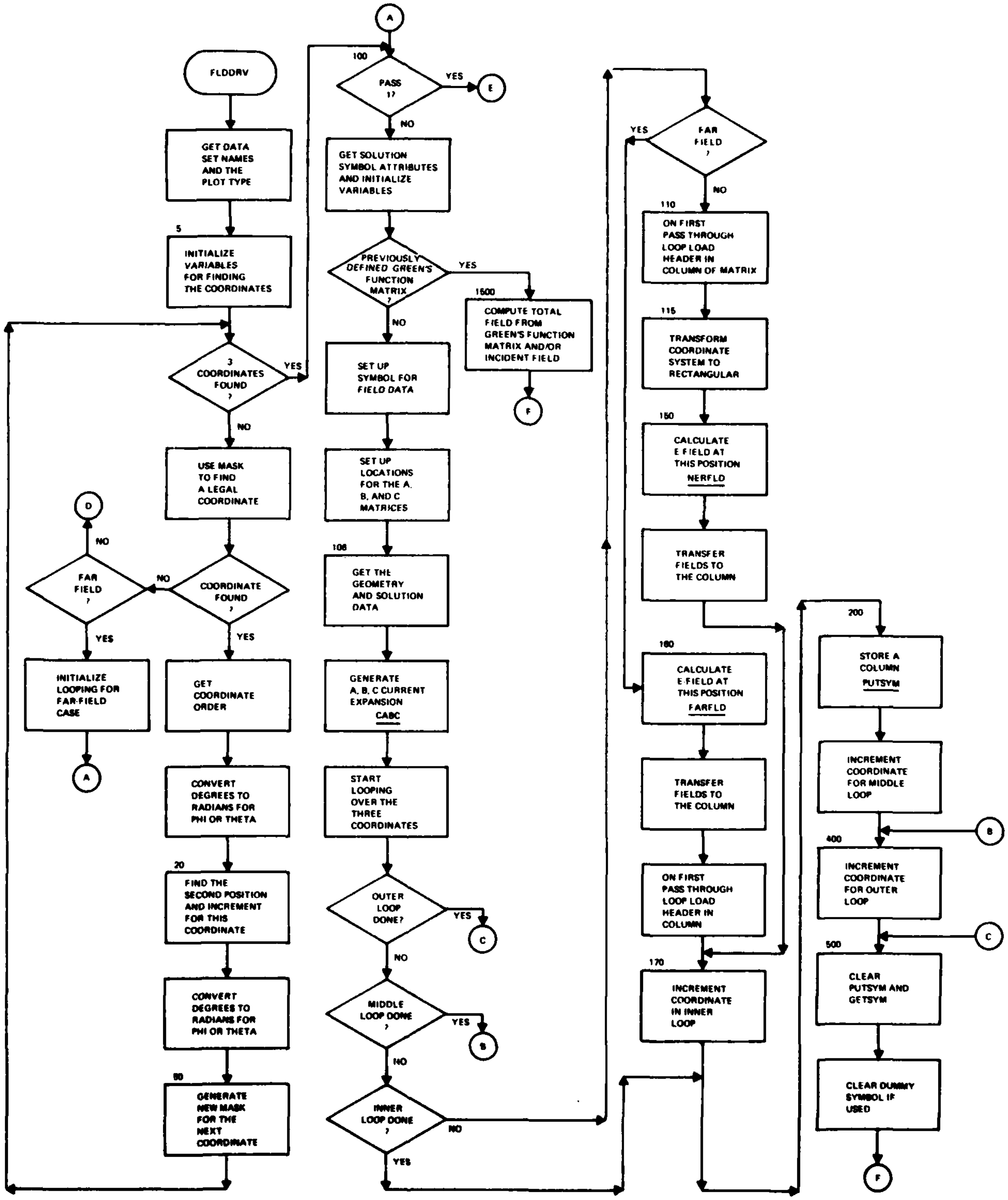
STATIN

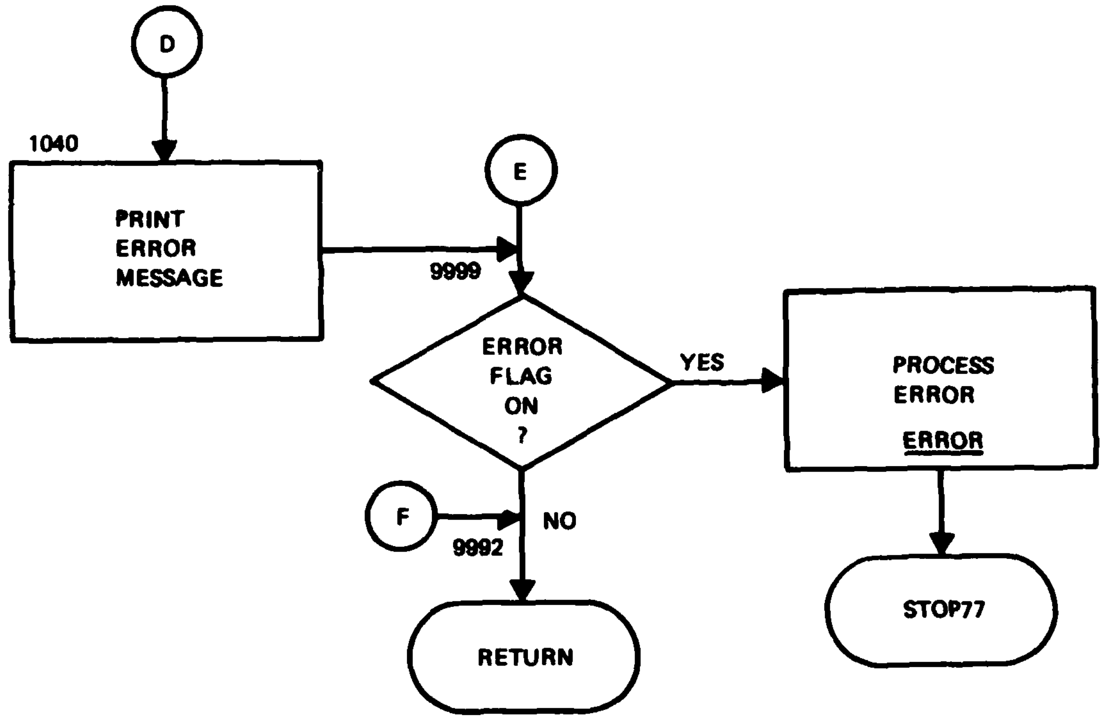
STATOT

SYMDEF

SYMUPD

WLKBCK


FLDDRV
GET DATA SET NAMES AND THE PLOT TYPE
5
INITIALIZE VARIABLES FOR FINDING THE COORDINATES
3 COORDINATES FOUND ?
YES
NO
USE MASK TO FIND A LEGAL COORDINATE
COORDINATE FOUND ?
FAR FIELD ?
D
INITIALIZE LOOPING FOR FAR-FIELD CASE
A
GET COORDINATE ORDER
CONVERT DEGREES TO RADIANS FOR PHI OR THETA
20
FIND THE SECOND POSITION AND INCREMENT FOR THIS COORDINATE
CONVERT DEGREES TO RADIANS FOR PHI OR THETA
50
GENERATE NEW MASK FOR THE NEXT COORDINATE
A
100
PASS 1?
E
GET SOLUTION SYMBOL ATTRIBUTES AND INITIALIZE VARIABLES
PREVIOUSLY DEFINED GREEN'S FUNCTION MATRIX ?
1500
COMPUTE TOTAL FIELD FROM GREEN'S FUNCTION MATRIX AND/OR INCIDENT FIELD
F
SET UP SYMBOL FOR FIELD DATA
SET UP LOCATIONS FOR THE A, B, AND C MATRICES
108
GET THE GEOMETRY AND SOLUTION DATA
GENERATE A, B, C CURRENT EXPANSION CABC
START LOOPING OVER THE THREE COORDINATES
OUTER LOOP DONE?
C
MIDDLE LOOP DONE ?
B
INNER LOOP DONE ?
FAR FIELD ?
110
ON FIRST PASS THROUGH LOOP LOAD HEADER IN COLUMN OF MATRIX
115
TRANSFORM COORDINATE SYSTEM TO RECTANGULAR
150
CALCULATE E FIELD AT THIS POSITION NERFLD
TRANSFER FIELDS TO THE COLUMN
160
CALCULATE E FIELD AT THIS POSITION FARFLD
TRANSFER FIELDS TO THE COLUMN
ON FIRST PASS THROUGH LOOP LOAD HEADER IN COLUMN
170
INCREMENT COORDINATE IN INNER LOOP
200
STORE A COLUMN PUTSYM
INCREMENT COORDINATE FOR MIDDLE LOOP
400
INCREMENT COORDINATE FOR OUTER LOOP
500
CLEAR PUTSYM AND GETSYM
CLEAR DUMMY SYMBOL IF USED
F


D
1040
PRINT
ERROR
MESSAGE
E
9999
ERROR
FLAG
ON
?
YES
PROCESS
ERROR
ERROR
F
9992
NO
RETURN
STOP77


FILM
3