COMPUTER PROGRAM DESCRIPTION - A LONG PERIOD ARRAY PROCESSING PACKAGE FOR ILLIAC IV

Ann Kerr, et al

Teledyne Geotech

Prepared for:

Air Force Technical Applications Center
Defense Advanced Research Projects Agency

11 October 1974

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1 REPORT NUMBER SDAC-TR-74-17	2. GOVT ACCESSION NO.	3 RECIPIENT'S CATALOG NUMBER AD/A-002969
4 TITLE *(and Subtitle)* COMPUTER PROGRAM DESCRIPTION - A LONG PERIOD ARRAY PROCESSING PACKAGE FOR ILLIAC IV		5. TYPE OF REPORT & PERIOD COVERED Technical 6 PERFORMING ORG. REPORT NUMBER
7 AUTHOR(s) Kerr, Ann and Wagenbreth, Gene		8. CONTRACT OR GRANT NUMBER(s) F08606-74-C-0006
9 PERFORMING ORGANIZATION NAME AND ADDRESS Teledyne Geotech 314 Montgomery Street Alexandria, Virginia 22314		10 PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11 CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency Nuclear Monitoring Research Office 1400 Wilson Blvd., Arlington, Va. 22209		12. REPORT DATE 11 October 1974 13. NUMBER OF PAGES 97
14 MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)* VELA Seismological Center 312 Montgomery Street Alexandria, Virginia 22314		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

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)*

18 SUPPLEMENTARY NOTES

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

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

This document describes a preliminary version of a long period array processing package designed around the FKCOMB algorithm for use on the ILLIAC IV computer. FKCOMB is a general-purpose array-processing program that uses frequency-wavenumber analysis to produce a bulletin which lists signal detections and various statistics for each detection. Two data editing and reformatting modules prepare the seismic data for FKCOMB and can be modified for use with other seismic algorithms. Preliminary refor-

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

matting of the seismic data is performed by DEM1. The data is edited and fast fourier transformed by DEM2.

The input parameters required for operating these programs and their subroutines are described in this document.

COMPUTER PROGRAM DESCRIPTION
A LONG PERIOD ARRAY PROCESSING PACKAGE FOR ILLIAC IV

SEISMIC DATA ANALYSIS CENTER REPORT NO.: SDAC-TR-74-17

AFTAC Project No.:	VELA VT/4709
Project Title:	Seismic Data Analysis Center
ARPA Order No.:	1620
ARPA Program Code No.:	3F10
Name of Contractor:	TELEDYNE GEOTECH
Contract No.:	F08606-74-C-0006
Date of Contract:	01 July 1974
Amount of Contract:	$2,237,956
Contract Expiration Date:	30 June 1975
Project Manager:	Royal A. Hartenberger (703) 836-3882

P. O. Box 334, Alexandria, Virginia 22314

ABSTRACT

This document describes a preliminary version of a long period array processing package designed around the FKCOMB algorithm for use on the ILLIAC IV computer. FKCOMB is a general-purpose array-processing program that uses frequency-wavenumber analysis to produce a bulletin which lists signal detections and various statistics for each detection. Two data editing and reformatting modules prepare the seismic data for FKCOMB and can be modified for use with other seismic algorithms. Preliminary reformatting of the seismic data is performed by DEM1. The data is edited and fast fourier transformed by DEM2.

The input parameters required for operating these programs and their subroutines are described in this document.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

INTRODUCTION

The preliminary version of the long period array processing package is designed to demonstrate the feasibility of using the ILLIAC IV computer for seismic processing. It consists of three modules. Data editing module one (DEM1) reformats the long period data, Data editing module two (DEM2) edits and fast Fourier transforms (FFT) the data and FKCOMB performs the FKCOMB algorithm on the data. Each of these modules is structured to allow for expansion to include additional data formats.

Figure 1 is an overall flow of data from SDAC to the ILLIAC site. Currently the data is received at ILLIAC over the ARPANET via file transfer protocol (FTP) in the SDAC low-rate tape format. A tape drive is available at the ILLIAC site and could be used for data transfer. Output from the program is transmitted to SDAC via the ARPANET.

The channel component data is retrieved by DEM1 and stored by array identifier in time order. Minor timing errors are corrected. DEM2 edits the data for timing and data errors, transforms the data and stores the output in a disk file for input to the FKCOMB algorithm.

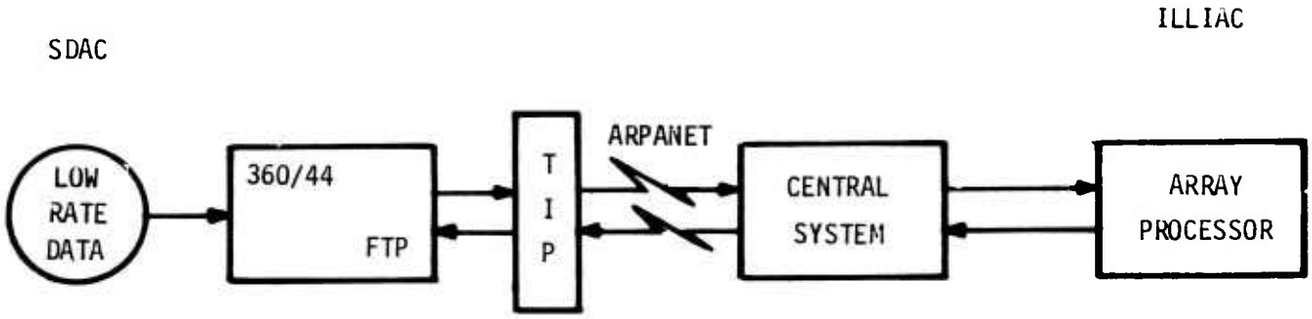


Figure 1. Data Path from SDAC to ILLIAC.

DATA EDITING MODULE 1 (DEM1)

PURPOSE

Seismic data as received from remote seismic arrays (LASA, NORSAR, ALPA) is not conveniently formatted for efficient retrieval by detection and analysis programs. As received, records from different arrays are randomly intermixed. The format and length of each record is dependent upon its site or origin. The density of data pertinent to most programs is under 20%. Over 90% of the runtime of the current version of FKCOMB is spent getting the data in a useful form. This problem is heightened by the parallel architecture of ILLIAC. In its raw form seismic data would be very difficult to manipulate in a parallel fashion. DEM1 is designed to read in raw data and organize it in a convenient manner for processing by such algorithms as FKCOMB. The structure of the data makes it difficult to maximize the processing power available on ILLIAC during such a reorganization. This is not important, as the process is inherently I/O bound. Utilization of the bandwidth between disk and core (.5*10**9 bits/sec.) and the large amount of core available (128K 64-bit words) makes it possible to do this job on ILLIAC quickly and makes the data available for storage on the UNICON laser memory for convenient access for any processing desired. The program is flexible and could handle data from all existing arrays and from any similar long period or short period arrays available in the future.

FUNCTIONAL AND THEORETICAL DISCUSSION

The input and output files used by DEM1 may be thought of as a series of 16-bit bytes. It is the function of DEM1 to retrieve the data bytes from the input file and organize them into the appropriate location in the output file. Other than some minor error correction, the order of the input bytes and the bytes themselves remain unchanged. Since the format of the input records is known, the task reduces to determining which byte of the input file is to be used to fill each byte of each output file, and having determined this to access the bytes in such a way to minimize time lost due

to memory and disk accesses. In practice, the method is turned around somewhat. Since the order of the input bytes is unchanged, it is effective to take bytes from the input file sequentially and insert each one into the next location in the appropriate output file. In order to conserve disk access time, large core buffers are used for input and output. ILLIAC core contains approximately 8×10^6 bits. Using buffers of 10^6 bits, 500 disk accesses are necessary to input 5×10^8 bits (24 hours of data) and another 500 disk accesses to output the data. Assuming the worst case of 40 millisecond access time this means that approximately 40 seconds of I/O time will be required to process 24 hours of data using a simple single buffering scheme for I/O, if all of core is used for buffering. In actuality, not all of core is available for I/O buffers and the time taken may be up to 80 seconds. Experimentation with different disk mappings to reduce the average access time may significantly reduce the time. The second major source of time is memory access. ILLIAC does not allow memory to memory transfers, so all data most go from memory to a register to memory. ILLIAC memory access time, though variable, due to overlap, is approximately 600 nanoseconds. If a transfer were done for each 16-bit byte, requiring two memory accesses, 2 accesses x (6×10^{-7}) seconds/access x 10^7 bytes = 12 seconds of memory access time would be required to transfer all input bytes. As coded, the scratch pad memory in the ILLIAC Control Unit (CU) is used to somewhat reduce this time.

The third major source of time is overhead due to the calculation of the address in each buffer before a transfer and the shifting necessary to coordinate the 16-bit byte size of the data with the 64-bit ILLIAC word size. The amount of time used in this process is estimated to be equal to or slightly greater than the time taken in memory accesses.

The time required by DEM1 to reorganize twenty four hours of raw long period seismic data is under two minutes of I/O time and under one minute of processor time. Allowing for unforeseen overhead and future code optimization, three to five minutes of total running time is a reasonable estimate for this system of data acquisition. More complicated and possibly more efficient algorithms suggest themselves but are deemed unnecessary due to the small savings in time possible.

PROGRAM DESCRIPTION

Seismic data, as received over remote lines is divided into records. Each record consists of a series of 16-bit bytes. While the length and exact format of each record depends upon the array at which it was created, the general format of each is:

Record Format

HEADER ID -	Identifies source array.
POSSIBLE TIMING WORD -	Present if one time word per record rather than one time word per scan (for definition of SCAN, see below). Format of time word dependent upon array.
VARIABLE LENGTH GAP -	Length determined by array.
MULTIPLE TIME SCANS -	Number dependent upon array. For format of SCAN, see below.
VARIABLE LENGTH GAP -	Length determined by array.

Time SCAN

VARIABLE LENGTH GAP-	Length determined by array.
POSSIBLE TIME WORD -	Present if time word per SCAN rather than time word per record. Format of time word dependent upon array.
MULTIPLE DATA CHANNELS -	Number and format dependent upon array.
VARIABLE LENGTH GAP -	Length determined by array.

In this general structure there are 10 variables which exactly determine the format of a record. They are:

1) HEADER ID
2) TIME WORD PER SCAN OR TIME WORD PER RECORD
3) FORMAT OF TIME WORD
4) LENGTH OF GAP BEFORE FIRST TIME SCAN
5) LENGTH OF GAP AT BEGINNING OF EACH SCAN
6) NUMBER OF DATA CHANNELS PER SCAN

7) FORMAT OF DATA CHANNELS

8) LENGTH OF GAP AT END OF EACH SCAN

9) LENGTH OF GAP AFTER LAST SCAN

10) NUMBER OF SCANS PER RECORD.

DEM1 determines the header ID from the first byte of a record and from this determines the source array. All the other values are then found in a table constructed to describe each array. New formats can be handled by adding to or modifying this table. The table is currently constructed at compile time via a data statement and is changed by recompiling one block data subroutine.

A pointer is maintained indicating which byte in the input stream is to be accessed next. Using this pointer, it is determined whether the data byte is on disk, in core, or in the CU scratch pad memory. The format of the input is accommodated by manipulating the pointer to space over gaps. In a series of nested loops, each record, then each time scan, then each data channel is handled. Output goes to one of six output buffers, then to one of six output files. Since only three seismic arrays are currently implemented, only three of these files are presently used. The others are available for expansion.

Timing errors in the input are of two types, gaps and reversals. Gaps of two or fewer are corrected by duplicating the data from the last good time step. This data is always available in the output buffer, though the duplication is complicated somewhat by the two levels of buffering used. Larger gaps are indicated by a diagnostic message and no correction is attempted. Time reversals are handled by ignoring any redundant data and a diagnostic message.

Processing continues until an unidentifiable input record is found. If the header is non-zero, a diagnostic is printed indicating the possibility of a data error. Normal end of data is indicated by a header of all zeroes. In either case, all output buffers are emptied to disk and processing terminates.

DATA AREAS AND SYMBOL DEFINITIONS

ADB - CU INTEGER - holds word from ADB buffer before being written in core.

ADBBUF(8) - CU INTEGER - CU scratch pad memory (ADB) input buffer.

ADBOUT(6) - CU INTEGER - one word ADB buffer for each array.

ADDRS - CU INTEGER - address in OUTBUF of IT.

ARRAY - CU INTEGER - Array currently being processed.

BCT - CU INTEGER - number of bytes in last partially filled word.

BYTCNT(6) - CU INTEGER - byte count for each output buffer.

BYTS - CU INTEGER - Number of bytes from last time step that are in core. Used in filling time gaps.

CNTRL (*,6) _ PE INTEGER - An array initialized at compile time giving the format of data from each seismic array.

DEBUG - CU INTEGER - Controls the output of debug print-out. Zero for typical run.

ENDADB - CU INTEGER - address of byte being held at end of ADB input buffer.

INBUF(*,128) - PE INTEGER - Input buffer.

INBYT - CU INTEGER - holds input byte after call to GETBYT.

INPTB - CU INTEGER - Pointer to current byte in input buffer.

INPTW - CU INTEGER - Pointer to current word in input buffer.

IT - CU INTEGER - first word of data to use when filling in time gaps.

LADB - CU LOGICAL - equivalenced to ADB to facilitate shifting and masking.

LADBBU(8) - CU LOGICAL - equivalenced to ADBBUF to facilitate shifting and masking.

LADBOU - CU LOGICAL - equivalenced to ADBOUT to facilitate shifting and masking.

LADBWR - CU LOGICAL - equivalenced to ADBWRD to facilitate shifting and masking.

LADDRS - CU LOGICAL - equivalenced to ADDRS to facilitate shifting and masking.

LARRAY - CU LOGICAL - equivalenced to ARRAY to facilitate shifting and masking.

LBCT - CU LOGICAL - equivalenced to BCT to facilitate shifting and masking.

LBYTCN(6) - CU LOGICAL - equivalenced to BYTCNT to facilitate shifting and masking.

LBYTS - CU LOGICAL - equivalenced to BYTS to facilitate shifting and masking.

LDEBUG - CU LOGICAL - equivalenced to DEBUG to facilitate shifting and masking.

LENDAD - CU LOGICAL - equivalenced to ENDADB to facilitate shifting and masking.

LINBYT - CU LOGICAL - equivalenced to INBYT to facilitate shifting and masking.

LINPTB - CU LOGICAL - equivalenced to INPTB to facilitate shifting and masking.

LINPTW - CU LOGICAL - equivalenced to INPTW to facilitate shifting and masking.

LIT - CU LOGICAL - equivalenced to IT to facilitate shifting and masking.

LORGCO - CU LOGICAL - equivalenced to ORGCOR to facilitate shifting and masking.

LOUBYT - CU LOGICAL - equivalenced to OUBYT to facilitate shifting and masking.

LOUPTW - CU LOGICAL - equivalenced to OUPTW to facilitate shifting and masking.

LPAGE - CU LOGICAL - equivalenced to PAGE to facilitate shifting and masking.

LTl - CU LOGICAL - equivalenced to TI to facilitate shifting and masking.

DATA AREAS AND SYMBOL DEFINITIONS

ADB - CU INTEGER	- holds word from ADB buffer before being written in core.
ADBBUF(8) - CU INTEGER	- CU scratch pad memory (ADB) input buffer.
ADBOUT(6) - CU INTEGER	- one word ADB buffer for each array.
ADDRS - CU INTEGER	- address in OUTBUF of IT.
ARRAY - CU INTEGER	- Array currently being processed.
BCT - CU INTEGER	- number of bytes in last partially filled word.
BYTCNT(6) - CU INTEGER	- byte count for each output buffer.
BYTS - CU INTEGER	- Number of bytes from last time step that are in core. Used in filling time gaps.
CNTRL (*,6) _ PE INTEGER	- An array initialized at compile time giving the format of data from each seismic array.
DEBUG - CU INTEGER	- Controls the output of debug print-out. Zero for typical run.
ENDADB - CU INTEGER	- address of byte being held at end of ADB input buffer.
INBUF(*,128) - PE INTEGER	- Input buffer.
INBYT - CU INTEGER	- holds input byte after call to GETBYT.
INPTB - CU INTEGER	- Pointer to current byte in input buffer.
INPTW - CU INTEGER	- Pointer to current word in input buffer.
IT - CU INTEGER	- first word of data to use when filling in time gaps.
LADB - CU LOGICAL	- equivalenced to ADB to facilitate shifting and masking.
LADBBU(8) - CU LOGICAL	- equivalenced to ADBBUF to facilitate shifting and masking.
LADBOU - CU LOGICAL	- equivalenced to ADBOUT to facilitate shifting and masking.

LADBWR - CU LOGICAL - equivalenced to ADBWRD to facilitate shifting and masking.

LADDRS - CU LOGICAL - equivalenced to ADDRS to facilitate shifting and masking.

LARRAY - CU LOGICAL - equivalenced to ARRAY to facilitate shifting and masking.

LBCT - CU LOGICAL - equivalenced to BCT to facilitate shifting and masking.

LBYTCN(6) - CU LOGICAL - equivalenced to BYTCNT to facilitate shifting and masking.

LBYTS - CU LOGICAL - equivalenced to BYTS to facilitate shifting and masking.

LDEBUG - CU LOGICAL - equivalenced to DEBUG to facilitate shifting and masking.

LENDAD - CU LOGICAL - equivalenced to ENDADB to facilitate shifting and masking.

LINBYT - CU LOGICAL - equivalenced to INBYT to facilitate shifting and masking.

LINPTB - CU LOGICAL - equivalenced to INPTB to facilitate shifting and masking.

LINPTW - CU LOGICAL - equivalenced to INPTW to facilitate shifting and masking.

LIT - CU LOGICAL - equivalenced to IT to facilitate shifting and masking.

LORGCO - CU LOGICAL - equivalenced to ORGCOR to facilitate shifting and masking.

LOUBYT - CU LOGICAL - equivalenced to OUBYT to facilitate shifting and masking.

LOUPTW - CU LOGICAL - equivalenced to OUPTW to facilitate shifting and masking.

LPAGE - CU LOGICAL - equivalenced to PAGE to facilitate shifting and masking.

LTI - CU LOGICAL - equivalenced to TI to facilitate shifting and masking.

LT2 - CU LOGICAL - equivalenced to T2 to facilitate shifting and masking.

LT3 - CU LOGICAL - equivalenced to T3 to facilitate shifting and masking.

LT4 - CU LOGICAL - equivalenced to T4 to facilitate shifting and masking.

LT5 - CU LOGICAL - equivalenced to T5 to facilitate shifting and masking.

LT6 - CU LOGICAL - equivalenced to T6 to facilitate shifting and masking.

LWORD - CU LOGICAL - equivalenced to WORD to facilitate shifting and masking.

LWORDS - CU LOGICAL - equivalenced to WORDS to facilitate shifting and masking.

LPRTIA - CU LOGICAL - equivalenced to PARTIAL to facilitate shifting and masking.

LSAVAD - CU LOGICAL - equivalenced to SAVADB to facilitate shifting and masking.

OLDTIM(*) - PE INTEGER - Time in deciseconds from the beginning of the year for current time step.

ORGCOR - CU INTEGER - identifies address within file of the first word in the core input buffer.

OTIMEA(6) - PE INTEGER - Saves OLDTIM when switching from one array to another.

OUBYT - CU INTEGER - passes output byte to subroutine PUTBYT.

OUPAGE(6) - PE INTEGER - page of current output file

OUPTWA(6) - PE INTEGER - Pointer to current word in output buffer for each array.

OUT(*,64,6) - PE INTEGER - Output buffer. Includes room for 6 seismic arrays.

OUTPTW - CU INTEGER - pointer to current word in output buffer.

PAGE - CU INTEGER - page to read from next.

PINT1(*) - PE INTEGER - Holds intermediate results throughout the program.

PRTIAL - CU INTEGER - Number of bits of IT to use when filling in time gaps.

SAVABD - CU INTEGER - Saves ADBOUT(ARRAY) while filling in time gaps.

SAVBCT - PE INTEGER - Saves BYTCNT (ARRAY) while filling in time gaps.

SAVPTW - PE INTEGER - Saves OUPTW while filling in time gaps.

SCANS - PE INTEGER - Number of time scans that have been read from the current record.

TIME(*) - PE INTEGER - Time in deciseconds from the beginning of the year for current time step.

TSTEPS(6) - PE INTEGER - Number of time steps that have been retrieved for each array.

T1 - CU INTEGER - holds intermediate results throughout program.

T2 - CU INTEGER - holds intermediate results throughout program.

T3 - CU INTEGER - holds intermediate results throughout program.

T4 - CU INTEGER - holds intermediate results throughout program.

T5 - CU INTEGER - holds intermediate results throughout program.

T6 - CU INTEGER - holds intermediate results throughout program.

WORD - CU INTEGER - used to build output word while filling time gaps.

WORDS - CU INTEGER - number of complete words that are in core from last time window. Used in filling time gaps.

LINKAGE

DEM1 calls subroutine PUTBYT, GETBYT, CNVTIM, and RDPRM which are described in the section on subroutines.

DISK AREAS

INPUT - binary input area. Format described in DEM1 program description.

OUPUT1, OUPUT2, OUPUT3, OUPUT4, OUPUT5, OUPUT6 - binary output areas. Each contains output from one seismic array. Currently the input consists of data from only three arrays, so only the first three areas contain data other than the header. The format of each area is:

HEADER - beginning of each area.

WORD1: Array ID.

WORD2: number of time steps in this disk area.

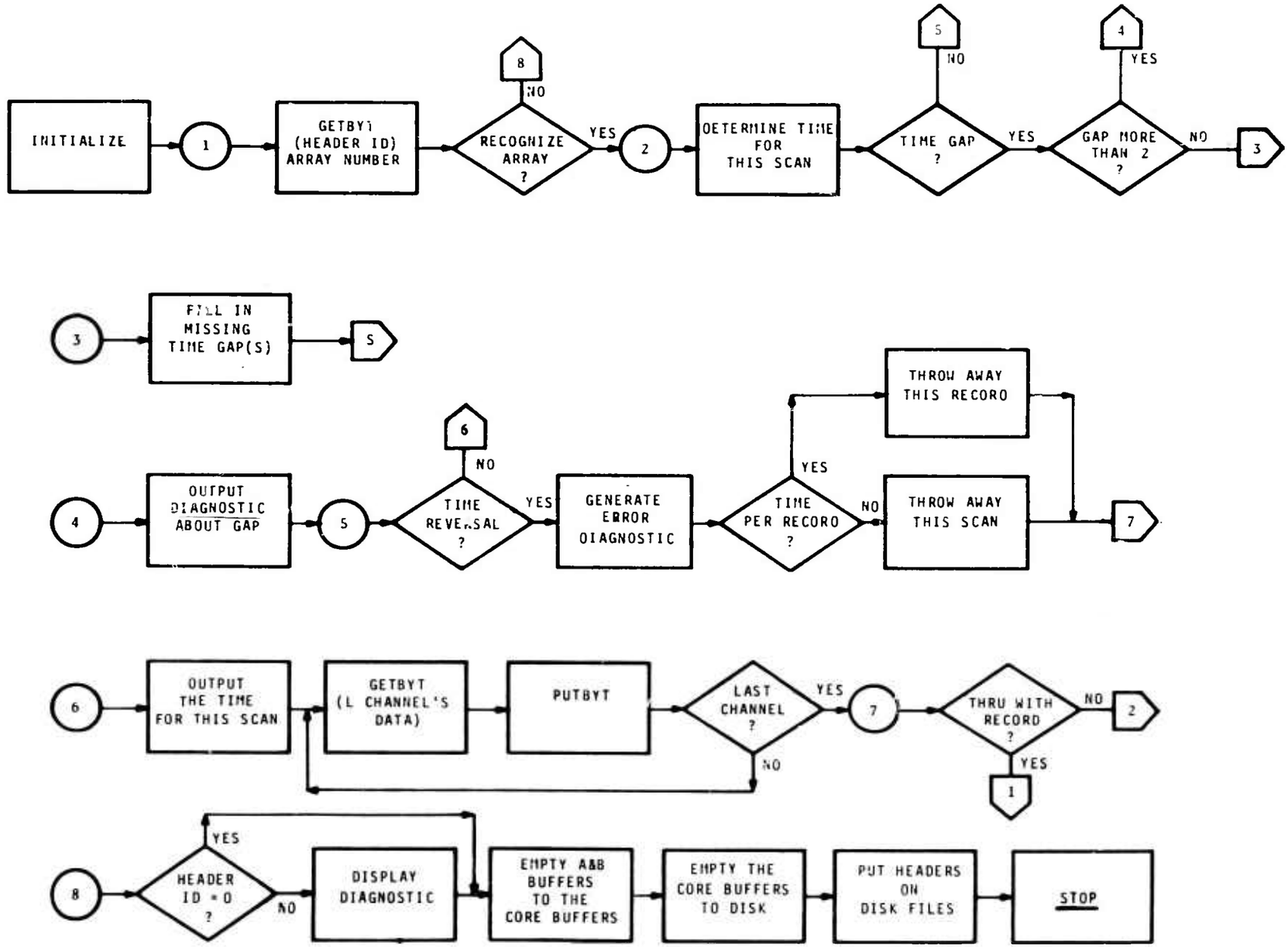
WORD3-1024: zero

TIME STEPS - first time step starts at word 1025. Number of time steps given in word 2 of the HEADER.

WORD1: time in deciseconds from beginning of year for this time step.

WORD2-N: data from each of N-1 channels.

DISPA - Print output. Consists of a header identifying the run, bulletins indicating time gaps and reversals, and a trailer indicating normal termination of the run.



DEM1 Main Program Flowchart

DATA EDITING MODULE 2 (DEM2)

PURPOSE

The FKCOMB algorithm requires that the input data be rearranged, edited, FFT'ed, and rearranged again before detection can begin. For ease of design and because of core limitations, this part of the processing has been made a separate module. DEM2 accepts the output of DEM1 and creates an output file which is error free and ready for FKCOMB detection. It handles any overlap requested and prints a bulletin whenever time gaps or data errors are detected. The data is deglitched and dead or noisy channels are removed.

FUNCTIONAL AND THEORETICAL DISCUSSION

Long period seismic data transmitted to SDAC is multiplexed by time. This organization is unaffected by DEM1. Prior to FKCOMB analysis, the following steps must be performed on the data:

- demultiplex by channel and segment into time windows
- overlap of time windows according to user specification
- convert to ILLIAC floating point format
- correct large time gaps by shifting of time windows
- deglitch
- remove dead and noisy channels
- fast fourier transform
- demultiplex by frequency.

Both steps involving demultiplexing are done a byte at a time. All other steps make full use of the parallel structure of ILLIAC by moving or computing 64 pieces of data at once.

The correction of time gaps and the overlapping of time windows involves moving data across processing elements (PE). Due to the organization of data into time windows whose size is a power of two equal to or greater than 64, this is an efficient use of the ILLIAC route instruction.

Spikes are smoothed via the following formula. Let An-1, A_n and A_{N+1} be three consecutive data elements and G be the user supplied "glitch" factor". If $|(A_n - A_{n+1})| > |G(A_{n-1} - A_{n+1})|$ then $A_n = (A_{n+1} + A_{n-1})/2$. Dead and noisy channels are detected via the mean square. If C_n is the mean square of the nth channel, MS is the mean square of all the channels and V is the user supplied "variance factor", a check is made to see if $1/V < C_n/MS < V$. If not, the channel is assumed to be dead or noisy.

PROGRAM DESCRIPTION

Raw seismic data and data as output by DEM1 are arranged by time step in the following manner:

T(1)[CH(1),CH(2),...,CH(N)], T(2)[CH(1),CH(2),...,CH(N)],...,
T(M)[CH(1),CH(2),...,CH(N)]

where "N" is the number of channels of the array being processed and "M" is the number of time steps contained in the data set being processed. As output by DEM1, the data is packed four 16-bit bytes per ILLIAC word. The first step of DEM2 is to read in a time window of data and multiplex it by channel so that it is arranged as follows:

CH(1)[T(1),T(2),...,T(W)],...,CH(N)[T(1),T(2),...T(W)]

where "N" is again the number of channels of the array being processed and "W" is the time window length, a power of two between 64 and 512 specified by the user at run time. During this process, the data is unpacked to one 16-bit byte per ILLIAC word. A double buffering scheme is used so that the last time window craeated is always accessible so that any overlap specified by the user can be handled and any time gaps recovered from, is possible.

The data is then deglitched and bad channels are noted. The data is then FFT'ed. The FFT routine is one written at the University of Illinois. It is necessary to convert the data to 32-bit floating point format before the routine is called. The FFT is performed in place and the output is a complex number stored in 32-bit form the inner and outer parts of the 64-bit word. The data is left in this form for use by FKCOMB. The last step is to

select the frequencies requested, rearrange the data so that each PE memory contains one complete multi-channel time window multiplexed by frequency. 640 words in each PE are reserved for the output, so the number of channels multiplied by the number of frequencies is restricted to less than 640. Since in normal operation fewer than 20 frequencies are requested, this poses no serious restriction.

DEM2 VARIABLES

ABUFF2(70400) - PE INTEGER - equivalenced to BUFF2.

ADBBUF(8) - CU INTEGER - ADB input buffer.

ADBWRD - CU INTEGER - current word in ADBBUF.

ALLMSQ(*) - PE REAL - the mean square of all channels for current time window.

ARRAY - CU INTEGER - indicates seismic array being processed. 1 = LASA
2 = ALPA
3 = NORSAR.

BF3PE - CU INTEGER - PE to be filled next with a time window.

BUFF2(*,500,2) - PE REAL - intermediate data buffer in which time windows are built.

BUFF3(8,640) - PE REAL - output buffer.

BYTE - CU INTEGER - byte within current ADB word most recently acquired.

CH - CU INTEGER - channel currently being processed.

CHGOOD(80) - PE INTEGER - indicates which channels passed variance test.

CHMSQ(80) - PE REAL - mean square for each channel.

CNTRL(*,6) - PE INTEGER - via assembled in data, give information on each array such as how many sensors.

COMP(*) - PE INTEGER - component of motion to be processed.
0 - vertical
1 - horizontal.

COREPT - CU INTEGER - gives the byte number of the first byte in BUFF1.

DEBUG - CU INTEGER - controls debug print out. For routine runs equals zero.

DIFFR - CU INTEGER - used in movement of data within time windows. Difference in rows.

DIFFW - CU INTEGER - used in movement of data within time windows. Difference in words.

F - CU INTEGER - frequency currently being processed.

FINSCN(*) - PE INTEGER - number of time scans in input.

GAP - CU INTEGER - non-zero if a time gap was found before this time window.

GLCHFT(*) - PE REAL - user supplied factor used in detecting glitches, or spikes, in the data.

HIFREQ - PE INTEGER - highest frequency processed.

IBUFF1(4096) - PE INTEGER - input buffer. Equivalenced to NBUFF1.

IBUFF3(*,640) - PE INTEGER - output buffer. Equivalenced to BUFF3(*,640).

INBYT - CU INTEGER - current input byte.

INDEX1 - CU INTEGER - holds intermediate results thru out program.

INDEX2 - CU INTEGER - holds intermediate results thru out program.

INDEX3 - CU INTEGER - holds intermediate results thru out program.

INDEX4 - CU INTEGER - holds intermediate results thru out program.

IPAGE - CU INTEGER - input area page number to read from next.

LCH - CU LOGICAL - equivalenced to CH to facilitate shifting and masking.

LDIFFW - CU LOGICAL - equivalenced to DIFFW to facilitate shifting and masking.

LF - CU LOGICAL - equivalenced to F to facilitiate shifting and masking.

LGAP - CU LOGICAL - equivalenced to GAP to facilitate shifting and masking.

LINBYT - CU LOGICAL - equivalenced to INBYT to facilitate shifting and masking.

LNEW - CU LOGICAL - equivalenced to NEW to facilitate shifting and masking.

LNGDCH - CU LOGICAL - equivalenced to NGDCH to facilitate shifting and masking.

LNGDR - CU LOGICAL - equivalenced to NGDR to facilitate shifting and masking.

LNGDST - CU LOGICAL - equivalenced to NGDST to facilitate shifting and masking.

LNGT - CU LOGICAL - equivalenced to NGT to facilitate shifting and masking.

LOFFSE - CU LOGICAL - equivalenced to OFFSET to facilitate shifting and masking.

LOFREQ - PE INTEGER - lowest frequency processed.

LOLD - CU LOGICAL - equivalenced to OLD to facilitate shifting and masking.

LTSCAN - CU LOGICAL - equivalenced to TSCAN to facilitate shifting and masking.

LT1 - CU LOGICAL - equivalenced to T1 to facilitate shifting and masking.

LT2 - CU LOGICAL - equivalenced to T2 to facilitate shifting and masking.

LT3 - CU LOGICAL - equivalenced to T3 to facilitate shifting and masking.

LT4 - CU LOGICAL - equivalenced to T4 to facilitate shifting and masking.

LT5 - CU LOGICAL - equivalenced to T5 to facilitate shifting and masking.

LT6- CU LOGICAL - equivalenced to T6 to facilitate shifting and masking.

LT7 - CU LOGICAL - equivalenced to T7 to facilitate shifting and masking.

LTWSZR - CU LOGICAL - equivalenced to TWSZR to facilitate shifting and masking.

NBUFF1(*,64) - PE INTEGER - input buffer. Also referenced by the equivalenced variable RBUFF1(4096).

NCHAN - CU INTEGER - number of channels.

NEW - CU INTEGER - which half of BUFF2 is new. Either one or two.

NGDCH - CU INTEGER - number of good channels after variance check.

NGDR - CU INTEGER - number of good rows. NGDCH x TWSZR.

NGDST - CU INTEGER - number of good sites.

NGT - CU INTEGER - number of good times. NGDCH x TWSZ.

NROWS - GU INTEGER - number of rows occupied by data in BUFF2. Equal to NSITE x TWSZ/64.

OFFSET - CU INTEGER - used in index calculation.

OLD - CU INTEGER - which half of BUFF2 is old. Either one or two.

OPAGE - CU INTEGER - page in output area to be written to next.

OTIME(*) - PE INTEGER - time in deciseconds from the beginning of the year of previous data word.

OVLAP - CU INTEGER - overlap between time windows.

PEN(*) - PE INTEGER - PE number. Initialized in block area subroutine to be 1 in pe number one, 2 in pe number two, and 64 in pe sixty four.

PINT1(*) - PE INTEGER - used for intermediate results thru out program.

PINT2(*) - PE INTEGER - used for intermediate results thru out program.

PREAL1(*) - PE REAL - used for intermediate results thru out program.

PREAL2(*) - PE REAL - used for intermediate results thru out program.

RBUFF1(4096) - PE REAL - input buffer. Equivalenced to NBUFFI and IBUFFI.

RBUFF2(70400) - PE REAL - intermediate buffer in which time windows are constructed. Equivalenced to BUFF2 and ABUFF2.

SITES(80) - PE INTEGER - give physical site number for each site used for output.

SITEGD(80) - PE INTEGER - indicates which sites passed variance test.

TIME(*) - PE INTEGER - time in deciseconds from the beginning of the year of current data word.

TOTSCN(*) - PE INTEGER - number of time scans procesed so far.

TSCANS - CU INTEGER - number of time scans down in current time windows.

TVARFT(*) - PE REAL - holds variance factor as it is modified to pass 50% of channels.

TWSZ - CU INTEGER - time window size. Integer power of two between 64 and 512.

TWSZR - CU INTEGER - number of rows occupied by one time window. TSWZ/64.

TWTIME(*) - PE INTEGER - time in deciseconds from the beginning of the year of time window currently being prepared by each PE.

T1 - CU INTEGER - holds intermediate results thru out program.

T2 - CU INTEGER - holds intermediate results thru out program.

T3 - CU INTEGER - holds intermediate results thru out program.

T4 - CU INTEGER - holds intermediate results thru out program.

T5 - CU INTEGER - holds intermediate results thru out program.

T6 - CU INTEGER - holds intermediate results thru out program.

T7 - CU INTEGER - holds intermediate results thru out program

VARPT(*) - PE REAL - user supplied factor used in detection of dead or noisy channels.

LINKAGE

In addition to the system I/O routines, DEM2 requires several relocatable modules which make up the University of Illinois FFT routine. These are:

FFT.MAINREL
FFT.SCRAMBLEREL
FFT.TRANREL
FFT.CONSTANTSREL

The interface to these modules is provided by the subroutine RUNFFT and is described in the documentation of that subroutine.

DEM2 also calls subroutine GTDATA, C16T64, C64T32, and ROWSUM described in the section on subroutines.

DISK AREAS

INDM2 - binary input. Described in DEM1 DISK AREAS.

OUTDM2 - binary output. Format:

HEADER - beginning of area.

WORD 1: Array ID

WORD2-1024: zero

TIME WINDOW - this data is best viewed as a two dimensional matrix. Each column will be wholly contained by a PE and contains one time window. There are 64 columns, one per PE. This is repeated as many times as required to contain all time windows processed. The format of each column is:

WORD 1: time in deciseconds for this time window.

WORD 2: number of good channels.

WORD 3-27: physical number of each good channel.

WORD 28-640:

First Frequency (channels 1-N)

.

.

.

last Frequency (channels 1-N)

.

CONPRM - input parameters for this run, in binary.

WORD1: DEBUG. Controls debug printout. Normally zero.

WORD2: time window size. Integer power of two between 64 and 512.

WORD3: over lap. Integer between zero and time window size minus one.

WORD4: glitch factor. Normally ten. Floating point.

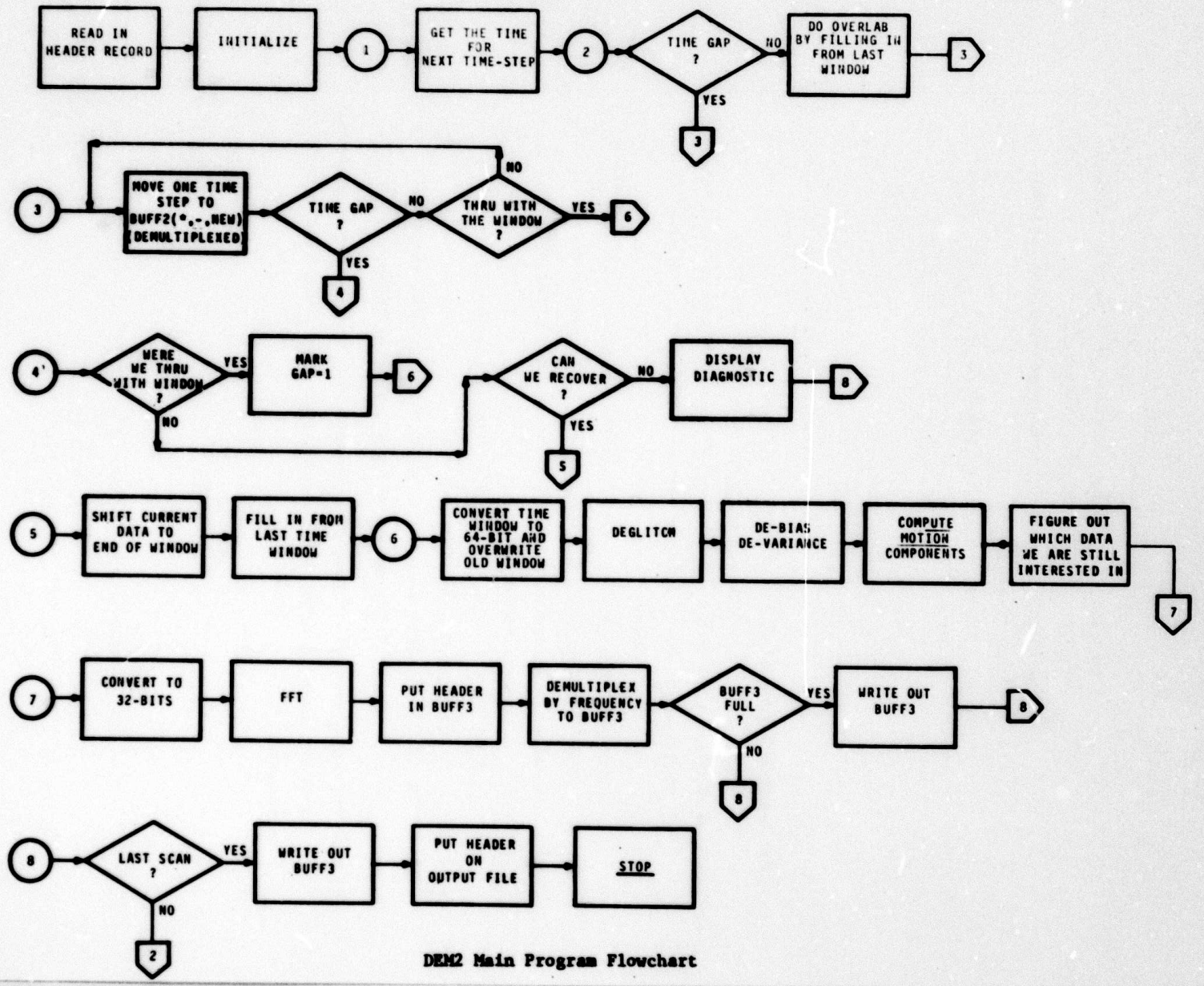
WORD5: variance factor. Normally ten. Floating point.

WORD6: low frequency. Integer indicating serial number of lowest frequency.

WORD7: high frequency. Integer indicating serial number of highest frequency.

WORD8: indicates whether vertical or horizontal components are to be processed. 1-vertical; 2-horizontal.

DISP2 - printed output consisting of a header identifying the run, information on each timing error located, and a trailer indicating normal termination.



DEM2 Main Program Flowchart

FKCOMB ALGORITHM

PURPOSE

To accomplish rapid signal detection using frequency beamforming, i.e., frequency wavenumber analysis, on a large number of beams.

FUNCTIONAL AND THEORETICAL DISCUSSION

Frequency-wavenumber (f-k) spectral estimation is a powerful technique for signal detection and waveform analysis of digitally recorded array data. The f-k spectrum of a given segment of array output is the squared modulus of the multidimensional Fourier transform of the data in time and space. The f-k spatial representation of a propagating wave is shown schematically in Figure 2. Using discrete Fourier analysis in the frequency domain, the representation can be thought of as a series of layers normal to the frequency axis, each layer representing the wavenumber plane at a given frequency. The wave is thus represented as a series of power maxima in the layers, and the locus of these maxima is determined by the phase velocity of the wave (Smart 1971).

The advantage of this process is that propagating wave components are easily recognized and separated from one another, subject to the limitations imposed by the array geometry, sensor weighting, and the type of spectral smoothing employed. In essence, f-k analysis is beamforming in the frequency domain. The method takes advantage of the fact that the signal-to-noise ratio varies with frequency, so the beamforming is done frequency by frequency. Also by staying in the frequency domain a great many beams can be examined rapidly, the number being limited only by the resolution cell of the array response. In practice this means that the azimuth and velocity of a signal need not be assumed: one merely accepts the beam with maximum power. This fact is important for signals such as long-period seismic surface waves, which not only are dispersive (i.e., their phase velocities vary with frequency) but whose arrival azimuth may also vary with frequency due to lateral inhomogeneties in their paths.

Frequency Wavenumber Representation

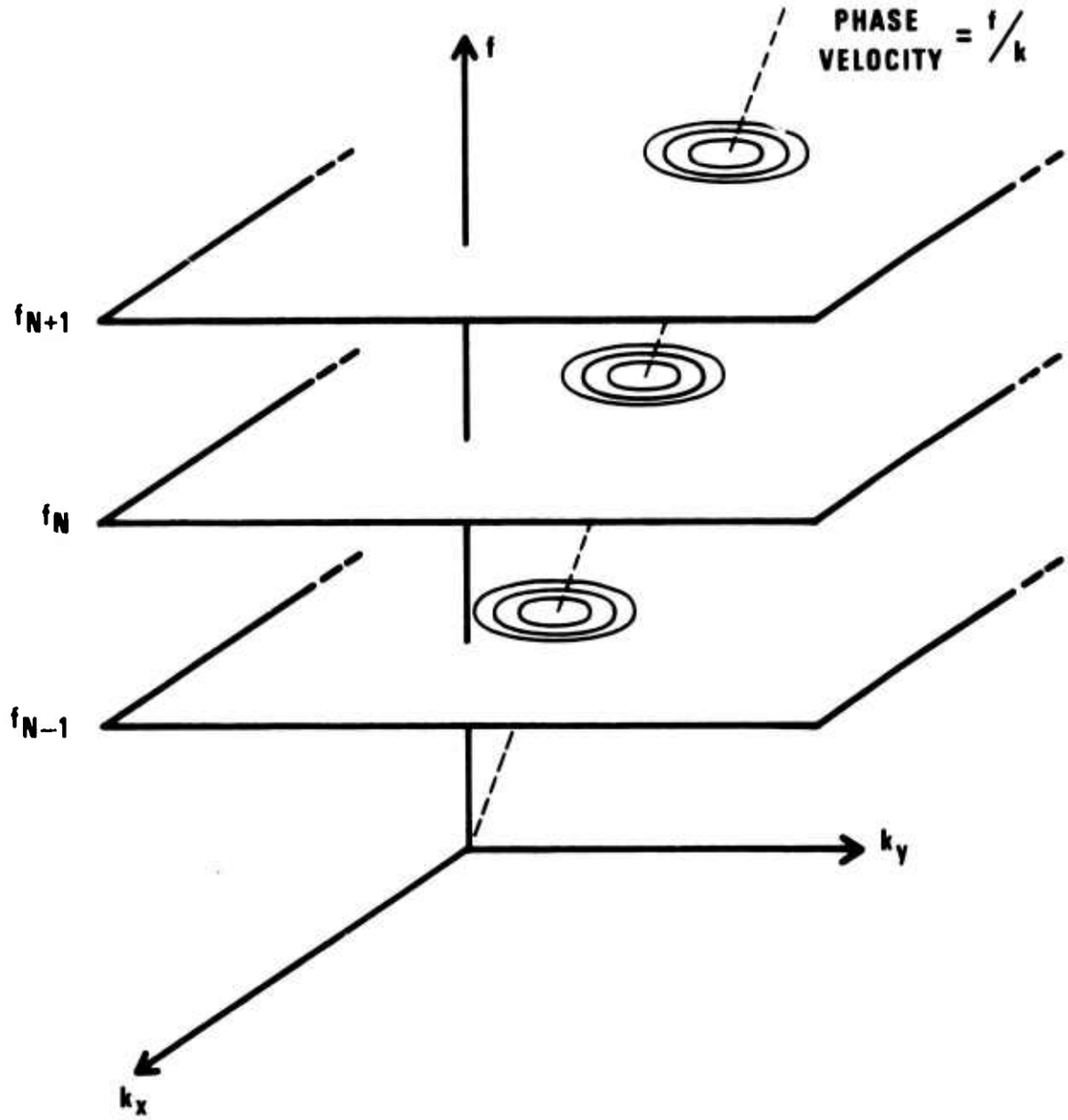


Figure 2. The frequency wavenumber representation of a propagating wave.

FKCOMB is a fast f-k analysis program that was used in an automatic processing system for microbarograph array data (Smart and Flinn, 1971).

Throughout the program description the Fourier transform of the nth user-specified channel component for any component of motion at angular frequency ω will be written as $A_n(\omega)e^{i\alpha}n(\omega)$.

PROGRAM DESCRIPTION

FKCOMB uses f-k analysis for continuous processing of time-varying data from arrays of sensors. Its output is in the form of a bulletin listing signal detections and giving for each the phase velocity, back azimuth, signal power, signal-to-noise ratio, and F statistic (related to signal-to-noise ratio; see Smart and Flinn 1971) as a function of frequency and arrival time.

FKCOMB examines time windows of data which have been Fourier-transformed in time and space. Maxima of power in three-dimensional f-k space are automatically picked and, if these maxima exceed a specified signal-to-noise detection threshold and are within a specified phase velocity range, they are listed together with their corresponding back azimuth, phase velocities, and other data. (See above.)

There are also two-dimensional maxima, which are places that are maximum within a given wavenumber plane but not along the frequency axis. If such two-dimensional maxima satisfy the specified signal-to-noise ratio and phase velocity criteria, and if their corresponding approximations to group velocity (see below) are reasonable, these maxima are also listed by the processor.

THE FAST f-k ANALYSIS ALGORITHM

The power at a given frequency and wavenumber is computed by means of equation (1):

$$P(f,\underline{k}) = \left|\frac{1}{N}\sum_{n=1}^{N} A_n(f)e^{i\alpha_n(f)}e^{2\pi i\underline{k}\cdot\underline{r}_n}\right|^2$$

where

f = frequency

k = vector wavenumber

N = number of channel components for the component of motion

n = channel component index

r_n = vector location of the n'th channel component with respect to an arbitrary origin

$A_n(f)e^{i\alpha_n(f)}$ = Fourier transform of the n'th channel component

$A_n(f)$ = amplitude part of the transform

$e^{i\alpha_n(f)}$ = phase part of the transform in which $\alpha_n(f)$ = the phase angle.

Equation (1) is evaluated for a matrix of wavenumber values at a series of discrete frequencies, as specified in the input parameters; it can be considered as a three-dimensional space with frequency being one dimension and the vector wavenumber $\underline{k}$ being the other two dimensions. For computation, $\underline{k}$ is resolved into a Cartesian coordinate system, with k_y related to geographic north and k_x related to geographic east.

A wavenumber value, say k_0, is related to the phase velocity V by equation (2):

$$V = f/|k_0|.$$

Stated verbally, phase velocity is inversely proportional to the distance from the frequency axis. The locus of constant values of V is a cone in f-k space, with the apex at the point $f = (k_x,k_y) = (0,0)$.

For f-k analysis the power is calculated in a matrix of wavenumber values separated by a grid interval Δk. This is greatly speeded up by using the relation shown in equation (3):

$$A_n(f)e^{i\alpha_n(f)}e^{2\pi i(k+k)\cdot r_n}$$

$$= A_n(f)e^{i\alpha_n(f)}e^{2\pi ik\cdot r_n}e^{2\pi i\Delta k\cdot r_n}$$

Thus if a set of N terms had been calculated for the first wavenumber value $\underline{k}_1$, the values at $\underline{k}_2 = \underline{k}_1 + \Delta\underline{k}$ are obtained by merely multiplying those terms by a factor $\exp(+2\pi i \Delta k \cdot r_n)$. Therefore, if a regular grid is used, only one set of kernels $\exp(+2\pi i \underline{k}_1 \cdot \underline{r}_n)$ need be generated, the remaining values being obtained with successive multiplication by the invariant kernels $\exp(+2\pi i \Delta\underline{k} \cdot \underline{r}_n)$.

THE SEARCH PROCESS

The frequency wavenumber search using triangular and rectangular grids can be thought of as taking place within a cone (Figure 3). At each frequency searched, the process takes place within a search disk bounded by the intersection of the cone and a constant frequency plane. This search disk is then extended by a border shaped like an annulus to insure complete coverage by the grid. Initially a triangular grid is used. Once a maximum is found, finer square grids are used, utilizing uphill walks from the maximum to get a better estimate. The program searches from lower to upper user-specified frequencies.

Beginning at the point of greatest power on the coarse grid, the program steps out in a plane of constant frequency along each of the four cartesian coordinate directions to determine the direction in which the power is rising, and it continues to compute successive points in that direction as long as the power is rising.

When the power begins to fall off in the direction being explored, a new direction is determined and followed, and the process is repeated until a place is reached where the four adjacent points in f-k space all show lower power. The grid spacing is then reduced by a factor of 6 and the same procedure is repeated to refine the location of the power maximum. The amount of computation required is about an order of magnitude less than would be required for computing and searching the complete two-dimensional spectral matrix.

All two-dimensional peaks located in this manner are then checked to see if they are also maxima in the frequency direction as well; such peaks are defined by the equation:

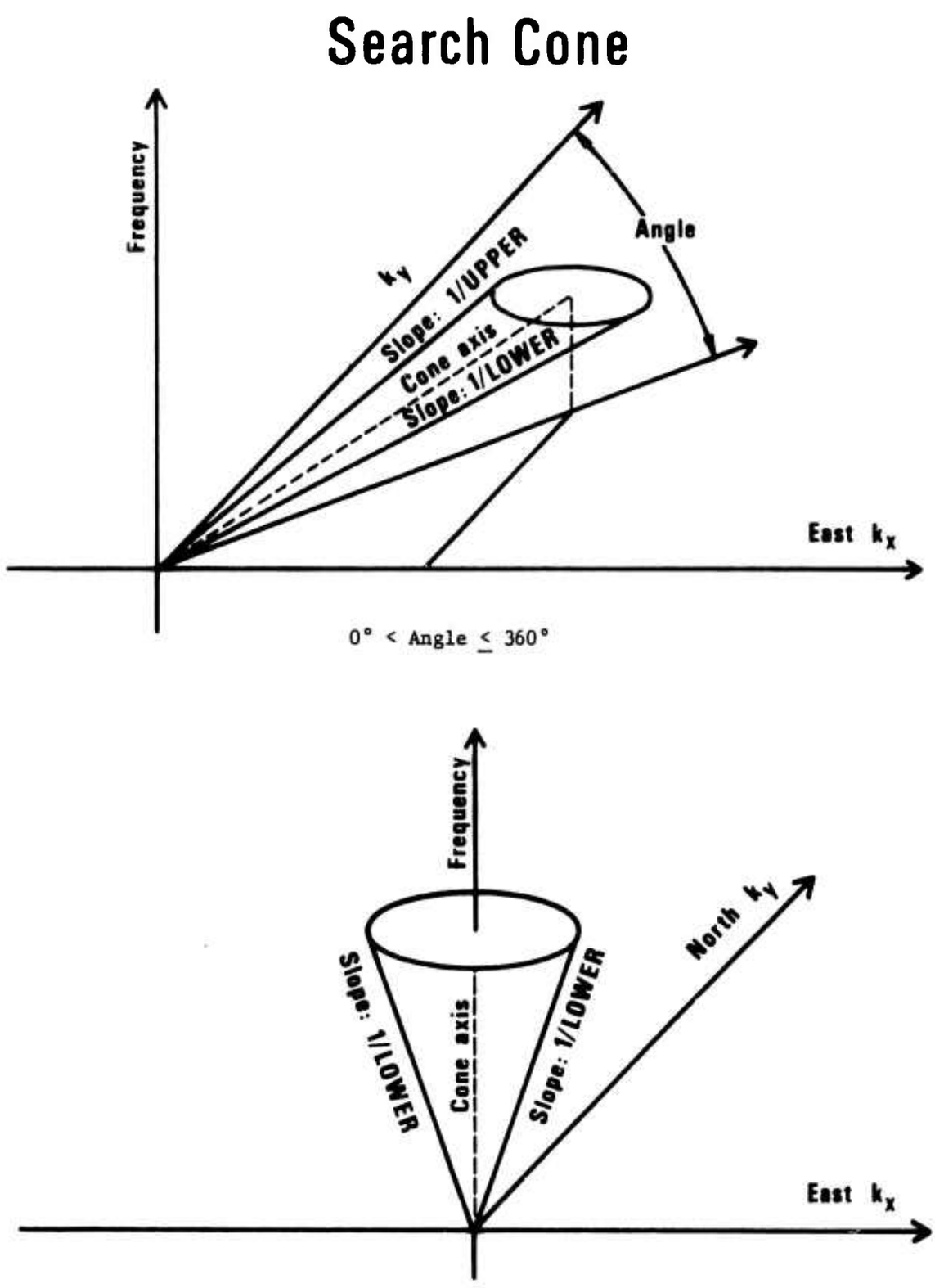


Figure 3. Angle = 0°

$$\frac{\partial P}{\partial f} = \frac{\partial P}{\partial k} = \frac{\partial P}{\partial k_y} = 0.$$

Of course the only extrema considered are the maxima, as minima are of no interest.

The program checks through the different frequencies as follows: consider the power in the wavenumber plane for the nth frequency. The power maximum in this layer is compared with the power and location of the maximum in the (n-1)th and (n+1)th layers; for clarity of exposition we call the power at these peaks P_n, P_{n-1}, and P_{n+1}. If both P_{n+1} and P_{n-1} are smaller than P_n, then P_n is flagged as a three-dimensional maximum. If either P_{n+1} or P_{n-1} is greater than P_n, a check is made on the respective positions of the peaks in wavenumber space; suppose that $P_{n-1} > P_n$. The vector wavenumber separation between the location of P_n and the location of P_{n-1} is calculated and the following logic is followed.

(a) If the separation is less than half the width of main lobe of the array response, i.e., within the detection cylinder, the two peaks are presumed to be part of the same signal, and P_n is not designated as a three-dimensional maximum.

(b) If the separation is greater than one half width of the main lobe of the array response, i.e., outside of the detection cylinder P_{n-1} is presumed to be part of another signal. To see where P_n itself is nevertheless a three-dimensional maximum, the power is checked in the (n-1)th and (n+1)th layers <u>at the same vector wavenumber as the peak P_n</u>: suppose these are Q_{n-1} and Q_{n+1} respectively. Then $P_n > Q_{n-1}$ and $P_n > Q_{n+1}$. This identification process is continued for all the frequency layers.

Wavenumber peaks which are two-dimensional but not three-dimensional, i.e., those for which

$$\frac{\partial P}{\partial k_x} = \frac{\partial P}{\partial k_y} = 0, \quad \frac{\partial P}{\partial f} \neq 0$$

may nevertheless appear in the bulletin, listed separately from the three-dimensional peaks. As a check on the validity of the two-dimensional peaks, a group velocity check is made in the following manner.

The values of $\Delta f/|\Delta k|$ are calculated for each peak and listed in the bulletin, where Δf is the frequency difference and Δk is the difference in wavenumber position of the maxima in the two adjacent layers. The ratio $\Delta f/|\Delta k|$ is a first-order approximation to the group velocity of a propagating signal; thus, if a two-dimensional peak is representative of the power in a propagating signal at that given frequency, the approximate group velocity should have a reasonable value.

A common cause of spurious two-dimensional peaks is leakage of power (due to the finite length of the time window) from a large peak in the spectrum. It has previously been shown (Smart, 1971) that this leakage occurs along lines of constant wavenumber (i.e., $\Delta k = 0$), so the calculated approximation to group velocity $\Delta f/|\Delta k|$ will fall outside the normal range of group velocities expected for seismic signals (2 to 3 kilometer/second).

At each frequency, only the first large two-dimensional maximum is considered in order to avoid picking up power entering the array through sidelobes in the array response.

FKCOMB at present picks only the primary peak at each frequency. Experience in using FKCOMB to analyze long-period seismic data has shown that signals are usually detected from the three-dimensional maxima, and the main interest in the two-dimensional peaks is to provide a more complete spectrum of the detected signal waveform.

DATA AREAS AND SYMBOL DEFINITIONS

By default in CFD, all CU variables are in common between all subroutines. All PE variables have been put in the common block FKMAIN. All variables may thus be treated as a global. This is the principle form of communication used between subroutines.

ADJF(*) - PE INTEGER	- used to hold index of adjacent frequency during 2-D or 3-D check.
ADKX(4) - PE REAL	- used in computing fine grid.
ADKY(4) - PE REAL	- used in computing fine grid.

ANGLE - CU REAL - input parameter indicating angle in degrees to be searched. Zero means search all directions.

ARRAY - CU INTEGER - array being processed.

AZ(*) - PE REAL - azimuth of signal at maximum power.

BORDER - CU REAL - width of border of search disk to insure that the disk is adequately covered by the coarse grid.

CHANAV(*) - PE REAL - used in calculation of signal to noise ratio.

CNTRL(*,6) - PE INTEGER - constant describing each array.

COSDK(*) - PE REAL - used in calculation of Re(Power)

COUNT2(*) - PE INTEGER - number of 2 dimensional maximum found.

COUNT3(*) - PE INTEGER - number of 3 dimensional maximum found.

DEBUG - CU INTEGER - controls generation of debug output.

DELTAF - CU INTEGER - difference between frequencies. SAMPLE RATE/TIME WINDOW SIZE.

DELTAY - PE REAL - coarse grid spacing in Y direction.

DELTAK - PE REAL - coarse grid spacing in X direction.

DELTX(500) - PE REAL - X displacement in K-space between successive points on the coarse grid.

DELTY(500) - PE REAL - Y displacement in K-space between successive points on the coarse grid.

DELX(*) - PE REAL - X-modification in K-space for direction being searched in fine grid search.

DELY(*) - PE REAL - Y-modification in K-space for direction being searched in fine grid search.

DIST - PE REAL - distance from X-axis (in K-space) to highest point on the coarse grid line being processed.

DKX - CU REAL - user input rectangular grid spacing.

DX(500) - PE REAL - K-space x-coordinate of each point on coarse grid.

DY(500) - PE REAL - K-space y-coordinate of each point on coarse grid.

FKX(*,25) - PE REAL - the x-coordinate in K-space of the power maximum for each frequency.

FFT(*,612) - PE REAL - equivalenced to INBUF91,28). Holds FFT output from DEM2.

FMAX(*,25) - PE REAL - the maximum power for each freqeuncy processed.

FPMAX(*) - PE REAL - power maximum for frequency currently being processed.

FSTAT(*) - PE REAL - fisher statistic.

HDKX - CU REAL - half of DKX.

HIFREQ - CU INTEGER - highest frequency to be considered.

I - CU INTEGER - used as a "DO" index throughout FKCOMB.

IGO - CU INTEGER - used to calculate point arrangement in lines of the coarse grid.

INBUF(*,640) - PE INTEGER - array into which the input data is read.

IND - CU INTEGER - "DO" index in subroutine GRID ranging from second to last coarse grid point.

INDEX - CU INTEGER - indicates frequency layer to check when determining 2 and 3-D maxima.

INFREQ - CU INTEGER - frequency currently being processed.

IPOWER(*,25) - PE REAL - Im(Power) of each channel for a point to be beamed from.

ITPOW(*) - PE REAL - Im(Power) for current point.

J - CU INTEGER - used as a "DO" index throughout FKCOMB.

K(*) - PE REAL - used in calculation of group velocity.

KERNEL(*,25) - PE REAL - holds intermediate results during power calculation.

KSEP(*) - PE REAL - separation in K-space of two adjacent power maxima.

KX - PE REAL - X-coordinate in K-space of center of search disk.

KY - PE REAL - Y-coordinate in K-space of center of search disk.

KXMAX(*) - PE REAL - X-coordinate in K-space of current power max.

KYMAX(*) - PE REAL - Y-coordinate in K-space of current power max.

KXSEP(*) - PE REAL - X part of separation in K-space of two adjacent power maxima.

KYSEP(*) - PE REAL - Y part of separation in K-space of two adjacent power maxima.

LINE - CU INTEGER - number of coarse grid lines on one side of the vertical axis.

LINEP1 - CU INTEGER - equal to one plus the number of lines on the coarse grid.

LINES - CU INTEGER - total number of lines on coarse grid equal to 2*lines+1.

LOCATE(*) - PE INTEGER - location of point on the coarse grid having maximum power.

LOC2D(*,25) - PE INTEGER - index indicating frequency in which each max was found.

LOC3D(8,25) - PE INTEGER - index indicating frequency in which each 3-D max was found.

LOFREQ - CU INTEGER - lowest frequency to be considered.

LOWER - CU REAL - user input limit bound on phase velocity.

MNCHAN - CU INTEGER - maximum number of channels in any PE.

MODE3 - CU LOGICAL - mode pattern indicating which PE's have detected a 3-D maximum.

N - CU INTEGER - used as a "DO" index throughout FKCOMB.

NCHAN(*) - PE INTEGER - the number of channels of data for each PE.

NFREQ - CU INTEGER - number of frequencies being processed.

NMODE - CU INTEGER - mode pattern indicating which PE's have ruled out a 3-D maximum.

NPOINT - CU INTEGER - index indicating which coarse grid point is being processed.

NPTS(*) - PE INTEGER - number of points on the coarse grid.

NTIMES - CU INTEGER - "DO" index controlling refinement loop in subroutine FNGRID.

OFFSET(*) - PE INTEGER - used in address calculation for access to INBUF.

PALE - CU INTEGER - next page to be read from input file.

PERIOD(*) - PE REAL - period of signal at power maximum.

PINT1(*) - PE INTEGER - holds intermediate results throughout FKCOMB.

PREAL1(*) - PE REAL - holds intermediate results throughout FKCOMB.

PREAL2(*) - PE REAL - holds intermediate results throughout FKCOMB.

RADIUS - CU REAL - radius of search disk.

REFINE - CU INTEGER - input parameter indicating number of times to refine fine grid.

RINBUF(*,640) - PE REAL - equivalenced to INBUF(1,1).

RPOWER(*,25) - PE REAL - Re(Power) of each channel for a point to be beamed from.

RTPOW(*) - PE REAL - Re(Power) for current point.

SAM - CU INTEGER - sampling rate.

SIGN - CU INTEGER - indicates sign of search direction on coarse grid.

SIGNAL(*) - PE REAL - signal to noise ratio.

SINDK(*) - PE REAL - used in calculation of Im(Power).

SWITCH - CU INTEGER - passes control information to subroutine grid.

TPOWER(*) - PE REAL - intermediate power maximum.

TWIN - CU INTEGER - time window length.

TWTIME(*) - PE INTEGER - time in deciseconds from beginning of year of beginning of time window being processed.

T1 - CU INTEGER - scratch variable used throughout FKCOMB.

T2 - CU INTEGER - temporary variable used throughout FKCOMB.

UPPER - CU REAL - user input limit bound on phase velocity.

VEL(*) - PE REAL - velocity of signal at maximum power.

X(*,25) - PE REAL - X-coordinate of channels used for each PE.

XCOORD(*) - PE REAL - X-coordinates of all channels.

Y(*,25) - PE REAL - Y-coordinate of channels used for each PE.

YCOORD(*) - PE REAL - Y-coordinates of all channels.

YMAX(50) - PE REAL - maximum Y displacement for each coarse grid line.

YPMI - CU INTEGER - one less than number of coarse grid points on coarse grid line under construction.

YPOINT(50) - PE REAL - number of points on each coarse grid line.

YTOP - CU INTEGER - maximum vertical offset of coarse grid line being processed.

LINKAGE

FKCOMB utilizes the standard COS, SIN, and SQRT functions supplied with CFD. It also utilizes the I/O package supplied by Institute for Advanced Computation (IAC).

FKCOMB also calls subroutine GRID, FNGRID, CHECKR, OUTPUT, REALE, IMG, and MAX which are discussed in the section on subroutine.

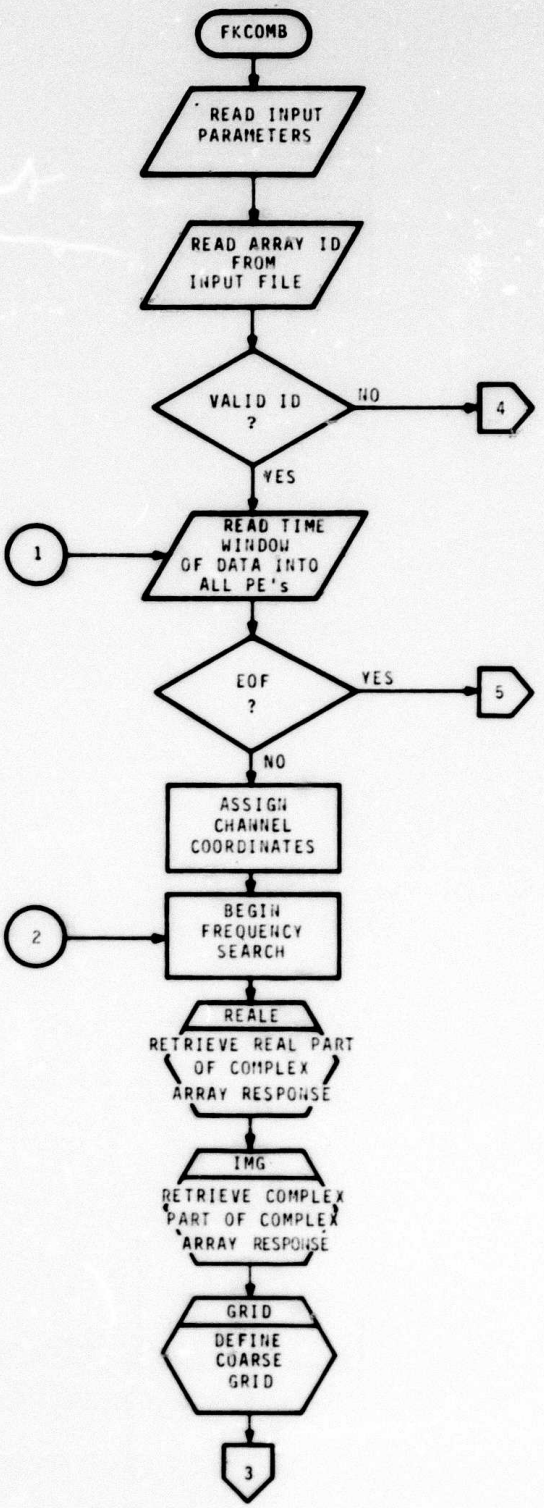
DISK AREAS

CONPRM - Contains user supplied run time parameters.

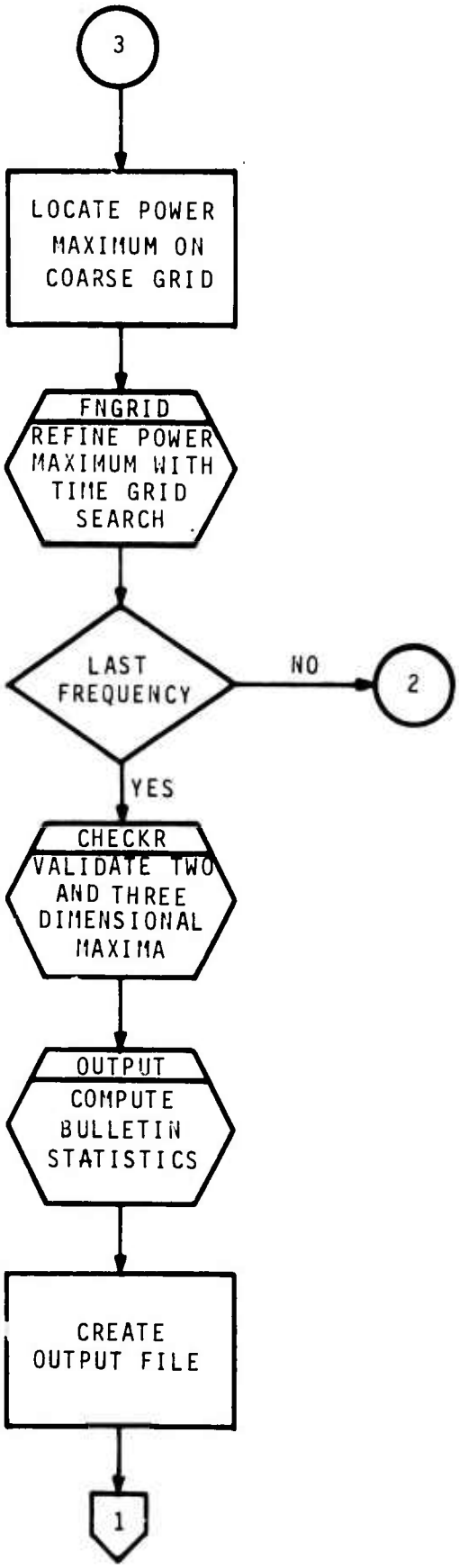
STCORD - Contains coordinates of sensors of array being processed.

FKIN - Main input area for FKCOMB. See output of DEM2.

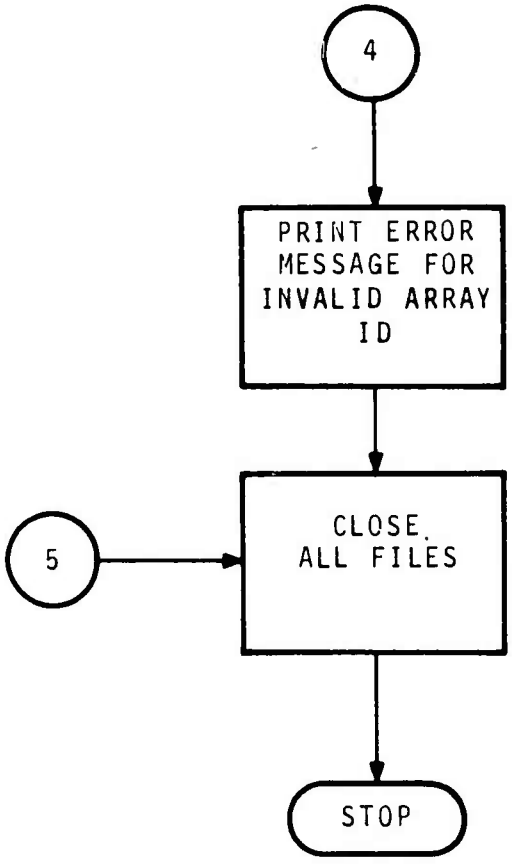
FKDISP - Area for printed output from FKCOMB.



FKCOMB Main Program Flowchart
Sheet 1 of 3.



FKCOMB Main Program Flowchart
Sheet 2 of 3.



FKCOMB Main Program Flowchart
Sheet 3 of 3

SAMPLE OUTPUT OF FKCOMB

The preliminary version of FKCOMB demonstrates the feasibility of processing seismic data on ILLIAC. The features and restrictions of this version were planned so that the code could be brought up in a minimum amount of time and still both demonstrate the feasibility of putting this and other algorithms on ILLIAC and provide a solid base for development of a version for routine processing of long period data. The code has been debugged to test sample data. Processing long period data routinely would require further debugging of the code and removal of the following restrictions:

Arrays - The current version has been tested on LASA data. Coding is included to handle ALPA and NORSAR data, but has not been tested. Processing these arrays differs only in DEM1 and DEM2; the FKCOMB analysis is identical.

Overlap - Overlapping of time windows is coded but untested in the current version. This capability requires the routing of data from one time window to the next and was not tested in the preliminary version.

Motion Components - The Gene Smart version of FKCOMB allows for processing of horizontal as well as vertical motion components. In actual use, vertical components are of primary interest. The processing of horizontal components was therefore designed into the algorithm, but never coded. If found to be necessary, this option can be added without major revision.

Channels - The ILLIAC version currently allows the user to specify the first channel and the last channel in the tape description in the block data subroutine initialization of the array CNTRL. This was thought to be sufficient. In practice it was found that it is often necessary to ignore arbitrary channels. This addition can be added quite simply by zeroing out any channel that is not used so that it does not affect the power maximum. It is still necessary to keep track of the number of real channels in order to correctly calculate average power and other statistics.

Variance (detection of dead and noisy channels) - As originally designed, after detection, dead and noisy channels were eliminated by actually removing them from the data stream. During implementation, especially after consideration of the removal of unwanted channels (see above), it was decided that a much better action after the detection of a dead or noisy channel is to zero it out and treat it exactly as an unwanted channel. Thus, the code for removal of the channel was never debugged and currently, though dead and noisy channels are detected, they are not removed. This addition requires the same effort as removing unwanted channels.

Stripping - Stripping is not included in the current version of FKCOMB, but is a straight forward addition. It is an option in the Smart version.

Half Frequencies - The Gene Smart version of FKCOMB zero fills before FFT in order to obtain twice as many frequencies. The added freqeuncy resolution gained is minimal, so this feature was not included in the ILLIAC version. In addition to the loss of some frequency resolution, there are several possible side affects which may result from this difference in algorithm between the two versions. In particular, the loss of resolution is most important for short time window lengths. Discrepancies between the results of the two versions must be analyzed before determining whether this feature should be added to the ILLIAC version. If found necessary, the addition is straight forward.

Timing - The run time is reducible by at least an order of magnitude by local optimization of code. Much of this consists of the insertion of assembly language code at critical points. The repeated calculation of exponentials in the analysis portion of the routine can be removed with little work and would result in a significant increase in speed. Due to these considerations, the lack of anything but wall clock time from ILLIAC, and the small amount of data currently being processed, it it impossible to give any timing data other than the theoretical time given elsewhere in this report.

The ILLIAC version of FKCOMB was tested by the comparison of it with the Gene Smart version currently available on the UCLA IBM 360/91. A seventeen minute and four second time period starting at 1628 GMT plus 12.0 seconds, day 79, 1972 was chosen for testing due to the existence of a large signal in the short period data from that time period. The following modifications were made to a standard deck used to run FKCOMB on the UCLA 360/91:

1. SLOP was changed from 4.0 to 1000.0 so that dead and noisy channels would not be eliminated so as to duplicate the conditions of the ILLIAC version.

2. Time window size and INC were both set to 256 to provide for a 256-point time window and no overlap.

3. Channel numbers and coordinates were modified so as to include channels 5 and 6, which normally are not used. This was necessary due to the restriction in the ILLIAC version discussed earlier. Channel coordinates of zero were used for these channels, as they are not valid long period channels.

The seismograms of an event in Honshu, Japan (t=15 57 50.4, 40.8N, 141.9E, March 19, 1972, m_b=6) chosen to compare the ILLIAC version of FKCOMB to the version programmed for the IBM 360/91 are shown in Figure 4. The position of four nonoverlapping 256 second time windows analyzed are also indicated on the figure. The time windows cover the initial portion of the Rayleigh wave train also including the arrival times of some seismic core phases (PKKP and PKKS). The results of parallel runs of the two versions are given in Figure 5. In the figure the results of the ILLIAC version are listed in the columns marked I4 and the results of the Smart version on the 360/91 are listed in the columns marked 91. The dashes indicate some events located by the ILLIAC version of FKCOMB were not listed by the Smart version of FKCOMB. The Smart version eliminates two dimensional maxima which do not lie along physically reasonable phase velocity lines through f-k space. This avoids reporting erroneous detections which are artifacts of the f-k spectral process. The sorting algorithm for review of results from FKCOMB analysis on ILLIAC will also eliminate these detections but was not implemented in the

preliminary version. The result agree well in general, the differences can be attributed mainly to differences in the details of search routines used. The velocities associated with the maximum F statistics in the tables are reasonably close to the phase velocities of Rayleigh waves at the corresponding periods, and the azimuths to those of the direction of the epicenter. Some of the values seem to be somewhat high but it must be kept in mind that the relatively small size of the LASA array does not permit the precise determination of phase velocities; thus, inaccuracies in phase velocities can be expected. The results indicate that the two programmed versions of FKCOMB work properly and yield comparable results.

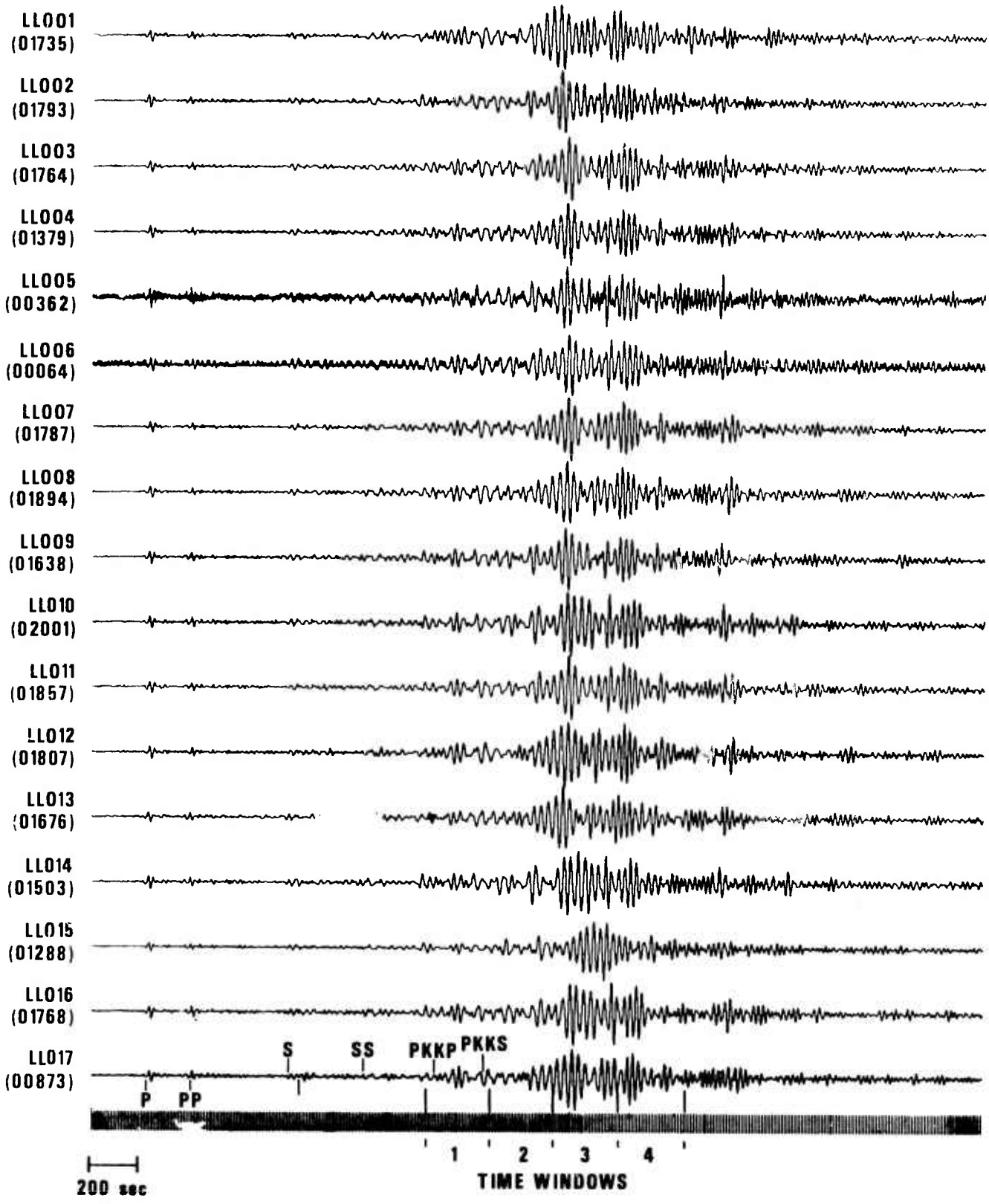


Figure 4. Seismograms of Honshu Event.

Time Window	Period(Sec)	Velocity(k/s)		Azimuth(Deg)		Power		Signal/Noise		Fisher Statistic		3d Max.	
1		I4	91	I4	91	I4	91	I4	91	I4	91	I4	91
079/162812	36.6	4.28	4.27	306	305	1352.	1350.	4.6	4.6	74	74	no	no
	32.0	6.76	5.63	307	309	157.7	157.	3.0	3.0	47	47	no	no
	28.4	6.45	5.01	311	316	1080.	1120.	2.1	2.3	33	37	no	no
	23.3	6.00	5.82	306	308	728.4	734.	2.9	3.0	47	48	no	no
	21.3	4.11	4.10	305	305	215.6	216.	1.0	1.0	17	17	no	no
	18.3	5.80	5.17	318	318	158.8	159.	1.4	1.4	22	22	no	no
	17.1	3.44	3.42	315	315	46.98	47.0	0.5	0.5	8	8	no	no
	15.1	3.43	–	296	–	31.84	–	0.6	–	10	–	no	–
	14.2	3.83	3.83	235	239	12.86	13.1	0.5	0.5	7	7	no	yes
	13.5	2.70	–	81	–	6.433	–	0.6	–	9	–	no	–
	11.6	3.28	3.51	180	182	3.295	3.68	0.4	0.4	6	7	no	yes
	10.7	22.98	10.47	325	16	1.267	1.45	0.4	0.5	6	8	no	no
	25.6	5.79	5.31	308	311	1265.	1300.	3.7	4.2	58	67	yes	yes
	19.7	6.27	6.15	322	322	411.7	412.	2.1	2.1	33	33	yes	yes
	16.0	3.69	3.51	296	297	52.47	52.6	0.7	0.7	11	11	yes	no
	12.8	3.07	3.06	50	48	6.946	7.00	0.4	0.4	6	6	yes	yes
	12.2	5.42	–	304	–	6.085	–	0.7	–	11	–	yes	–
	11.1	5.33	4.77	16	12	2.542	3.02	0.6	0.8	10	13	yes	yes
2													
079/163228	36.6	6.58	4.39	354	350	418.7	617.	0.4	0.7	6	11	no	no
	28.4	4.15	3.98	300	304	4052.	4210.	2.1	2.4	34	38	no	no
	25.6	4.71	4.10	301	308	1880.	2310.	1.7	3.5	28	50	no	no
	23.3	4.55	3.71	326	327	221.1	259.	0.6	0.7	9	12	no	no
	21.3	3.39	–	270	–	224.4	–	0.5	–	8	–	no	–
	18.3	3.78	–	267	–	121.1	–	0.6	–	9	–	no	–
	17.1	6.99		300	–	68.49	–	0.5	–	9	–	no	–
	15.1	7.15	–	293	–	41.38	–	1.1	–	18	–	no	–
	13.5	8.48	6.56	294	299	29.26	31.2	0.4	.4	6	7	no	–
	12.2	3.30	–	3	–	12.94	–	0.6	–	9	–	no	–
	11.6	10.22	10.11	300	304	11.61	11.9	0.6	0.6	9	10	no	yes
	11.1	11.24	10.07	294	296	8.08	8.06	0.8	0.8	13	12	no	no
	32.0	3.72	3.65	298	298	812.7	813.	4.2	4.2	67	67	yes	yes
	19.7	4.14	3.69	290	298	954.2	1290.	1.2	2.7	19	43	yes	yes
	16.0	2.97	–	42		43.86	–	0.4	–	7	–	yes	–
	14.2	8.64	7.29	296	301	51.70	54.7	0.8	0.9	12	14	yes	yes
	12.8	6.68	6.58	302	302	19.92	19.9	0.7	0.7	12	12	yes	no
	10.7	11.65	5.27	294	337	13.21	2.81	1.3	.1	20	2	yes	yes

Figure 5. Sample FKCOMB Output.
Sheet 1 of 2.

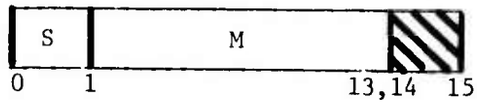
Time Window	Period(Sec)	Velocity(k/s)		Azimuth(Deg)		Power		Signal/Noise		Fisher Statistic		3d Max.	
3		I4	91	I4	91	I4	91	I4	91	I4	91	I4	91
079/163644	36.6	3.17	–	294	–	285.6	–	0.3	–	5	–	no	–
	32.0	3.99	3.60	302	311	3325.	3870.	1.7	2.7	27	43	no	no
	25.6	4.47	3.81	294	299	17820.	18500.	1.1	1.2	17	19	no	no
	23.3	3.76	3.77	304	304	6471.	6470.	2.4	2.4	39	39	no	no
	21.3	4.95	3.83	326	329	2374.	3340.	0.7	1.5	12	24	no	no
	19.7	3.72	3.39	300	294	638.3	653.0	0.9	1.0	15	16	no	no
	18.3	3.45	3.44	298	296	703.0	708.	0.6	0.6	10	10	no	no
	17.1	3.49	3.40	300	305	799.0	904.	1.2	1.7	20	27	no	no
	15.1	7.08	–	333	–	121.3	–	0.4	–	7	–	no	–
	14.2	3.21	3.14	20	18	39.32	40.2	0.5	0.5	7	7	no	no
	11.1	4.39	4.45	245	247	7.752	8.12	0.4	0.4	6	7	no	no
	28.4	4.19	3.73	320	320	47120.	47200.	3.8	3.8	60	61	yes	yes
	16.0	3.67	3.43	297	302	1221.	1280.	1.3	1.4	20	22	yes	no
	13.5	3.30	3.24	19	18	20.12	21.6	0.3	0.3	5	5	yes	no
	12.8	8.52	6.67	328	332	20.50	30.4	0.5	1.0	8	15	yes	no
	12.2	6.25	6.40	330	330	23.40	23.9	0.9	1.0	15	15	yes	no
	11.6	4.26	4.27	245	246	13.44	13.6	0.4	0.4	7	7	yes	no
	10.7	10.18	7.95	328	329	10.59	13.6	0.5	0.8	9	13	yes	no
4													
079/164100	32.0	3.42	–	329	–	200.3	–	0.4	–	7	–	no	–
	28.4	4.57	3.97	326	354	879.9	891.	1.6	1.6	25	26	no	no
	23.3	3.76	3.70	317	317	10730.	10700.	2.5	2.5	40	40	no	no
	21.3	3.89	3.68	314	319	8652.	9180.	1.7	2.1	28	33	no	no
	19.7	2.78	–	237	–	480.5	–	0.3	–	6	–	no	–
	17.1	3.40	3.13	297	299	284.6	290.	0.9	0.9	14	15	no	no
	15.1	3.85	3.62	317	319	332.5	354.	1.4	1.7	23	27	no	no
	14.2	4.09	3.83	314	314	132.5	133.	0.9	0.9	14	14	no	no
	13.5	4.47	4.46	314	314	68.17	68.1	0.6	0.6	10	10	no	no
	12.8	8.27	–	326	–	21.02	–	0.4	–	6	–	no	–
	11.6	8.60	7.16	321	322	22.64	24.0	0.9	1.0	14	16	no	no
	11.1	9.43	8.15	325	324	17.04	17.2	0.9	0.9	14	14	no	no
	10.7	5.75	3.76	316	335	26.59	6.26	0.7	0.2	12	4	no	yes
	36.6	6.28	4.81	347	349	429.9	481.	1.2	1.5	19	24	yes	yes
	25.6	3.92	3.53	321	320	16390.	16400.	4.5	4.5	71	71	yes	no
	18.3	3.63	3.59	302	302	1529.	1530.	2.1	2.1	34	34	yes	no
	16.0	3.84	3.63	308	309	1167.	1170.	2.4	2.4	39	39	yes	no
	12.2	4.93	4.93	315	315	29.63	29.6	0.6	0.6	10	10	yes	no

Figure 5. Sample FKCOMB Output.
Sheet 2 of 2

C16T64

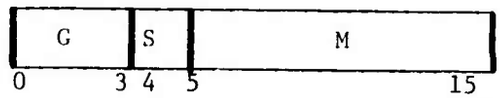
DESCRIPTION

C16T64 converts 16 bit raw data as sent from the seismic arrays to 64 bit ILLIAC floating point format. The format of the 16 bit data depends upon the value of ARRAY. If ARRAY is equal to 1 or 2, the data consists of a 14 bit two's complement integer left justified in a 16-bit byte. The data is extracted as shown:



$$VALUE = S \times(-2^{13})+M$$

If ARRAY is equal to 3, the data consists of a 12 bit two's complement number and a 4 bit gain code, all right justified. The data is extracted as follows:



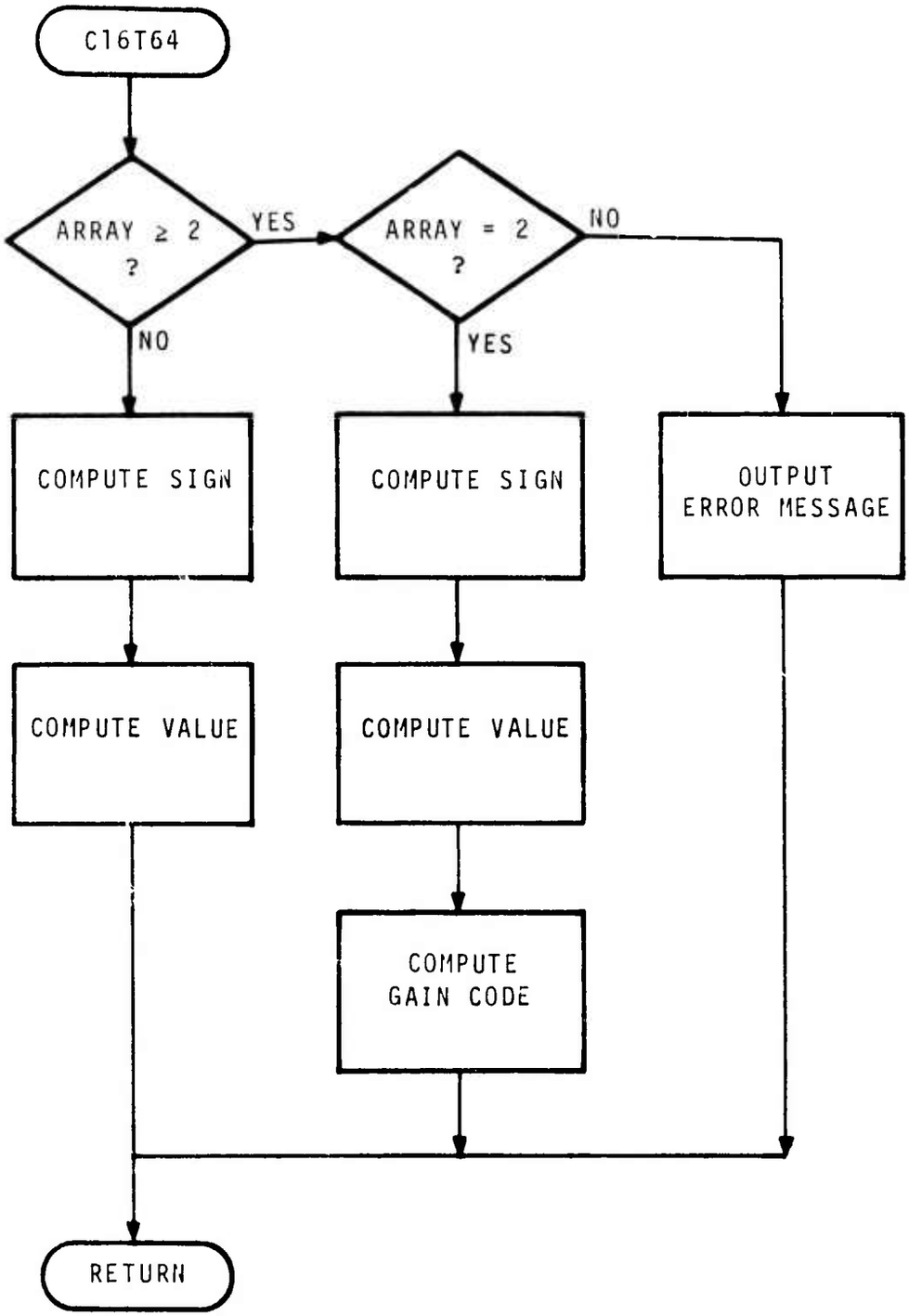
$$VALUE = (S \times(-2^{11})+M) \times 2^{10-G}$$

Most of the code consist of inserted assembly language code in a CFD driver. If ARRAY is not 1, 2 or 3, an error message is generated.

ARGUMENTS

IN(*) - PE REAL - input data.

OUT(*) - PE REAL - output data.


C16T64
ARRAY ≥ 2 ?
YES
ARRAY = 2 ?
NO
NO
YES
COMPUTE SIGN
COMPUTE SIGN
OUTPUT ERROR MESSAGE
COMPUTE VALUE
COMPUTE VALUE
COMPUTE GAIN CODE
RETURN

C64T32

DESCRIPTION

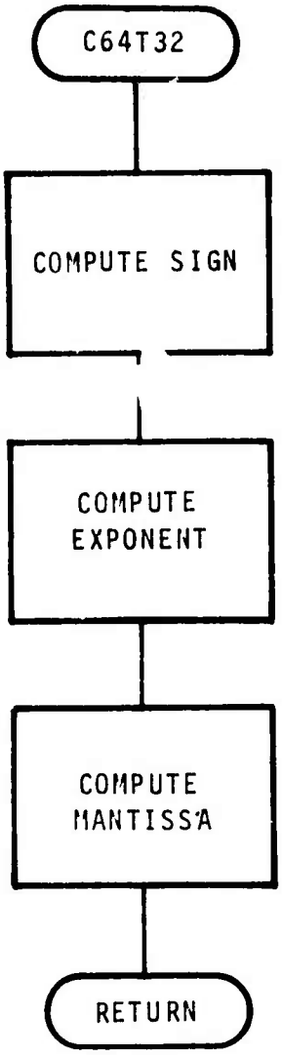
C64T32 creates a 32-bit ILLIAC floating point number from a 64-bit floating point number. The 32-bit result is left in the outer half of a 64 bit word. The sign is transferred directly. The mantissa is truncated without rounding to 24 bits. The new exponent is computed via the following formula:

NE = new exponent
OE = old exponent
NE = (OE-16384)+64

No check is made to see if the input number is outside the range of 32-bit representation. The inner half of the word is zero.

ARGUMENTS

IN(*) - PE REAL - 64-bit floating point input.
OUT(*) - PE REAL - 32-bit floating point output.


C64T32
COMPUTE SIGN
COMPUTE EXPONENT
COMPUTE MANTISSA
RETURN

CHECKR

DESCRIPTION

Subroutine CHECKR examines the power maximums found at each frequency to determine which maximums are three dimensional and which are two dimensional in frequency wavenumber space.

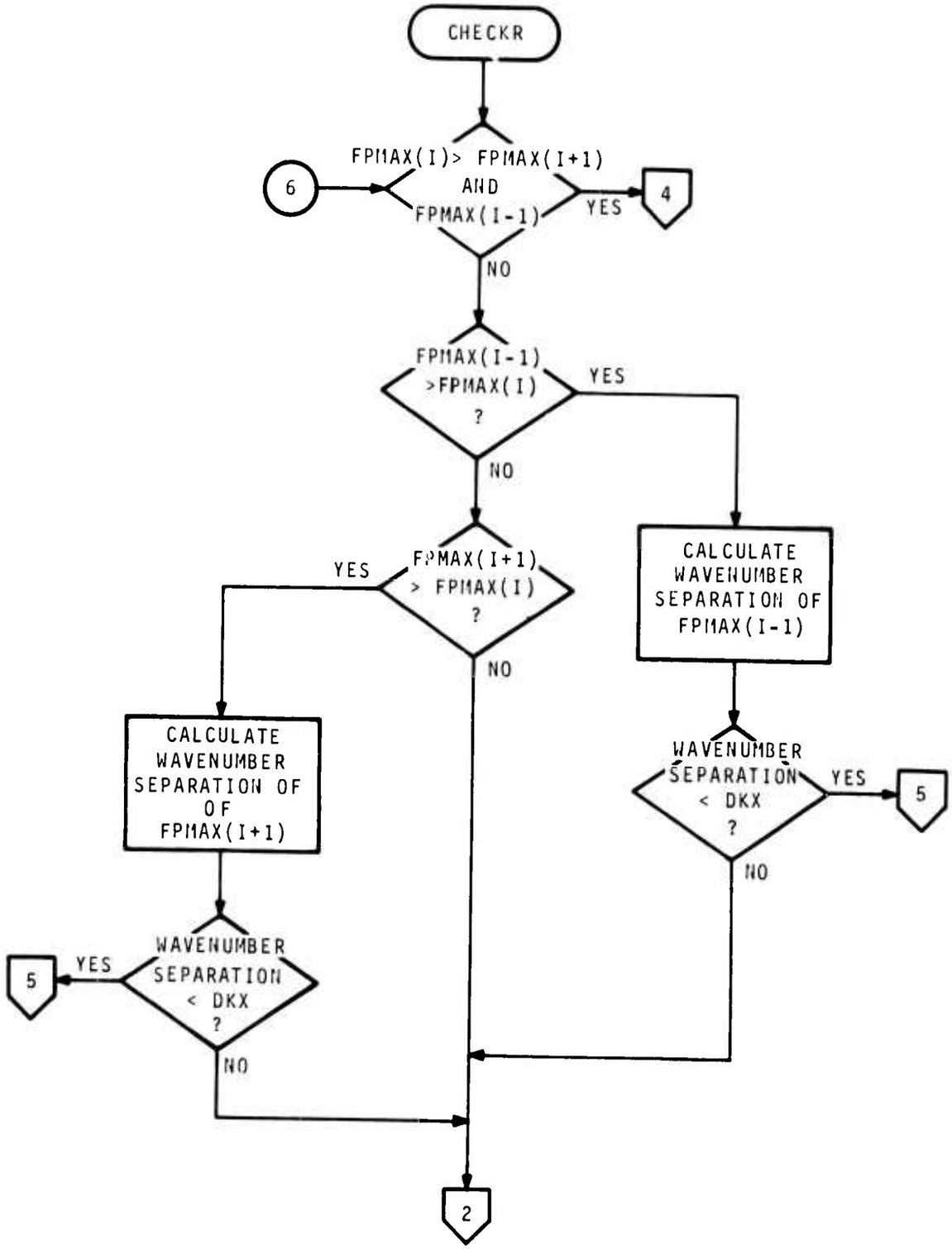
ARGUMENTS

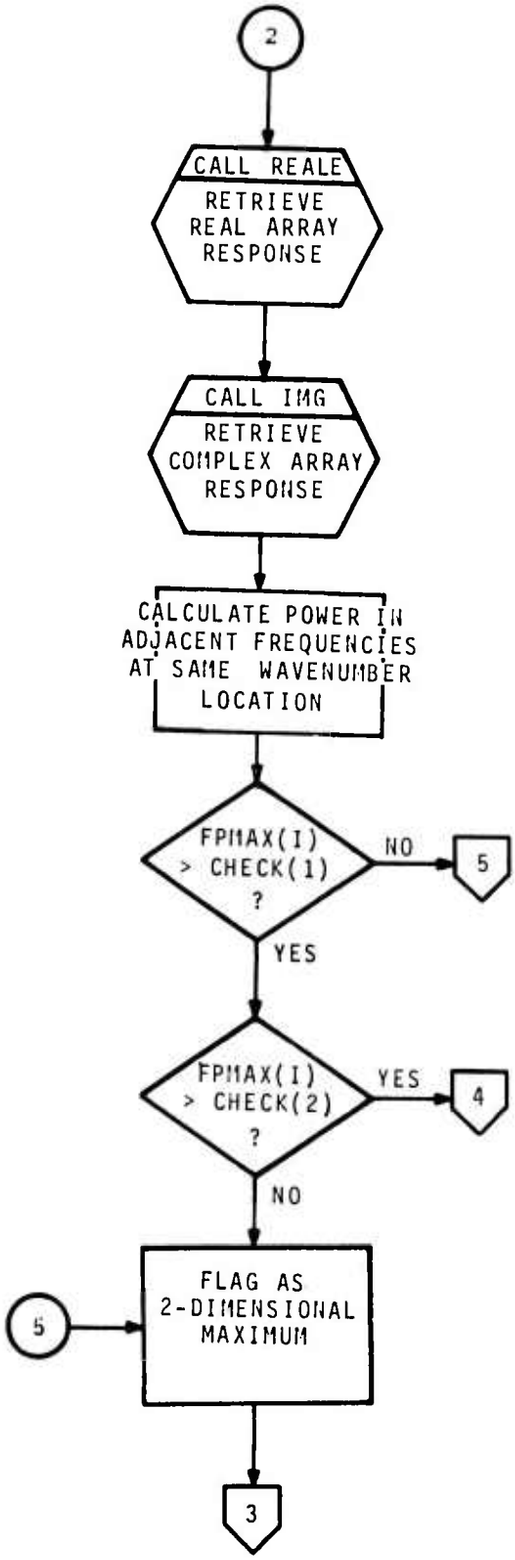
FPMAX(*) - PE REAL - power maximum for frequency being processed.

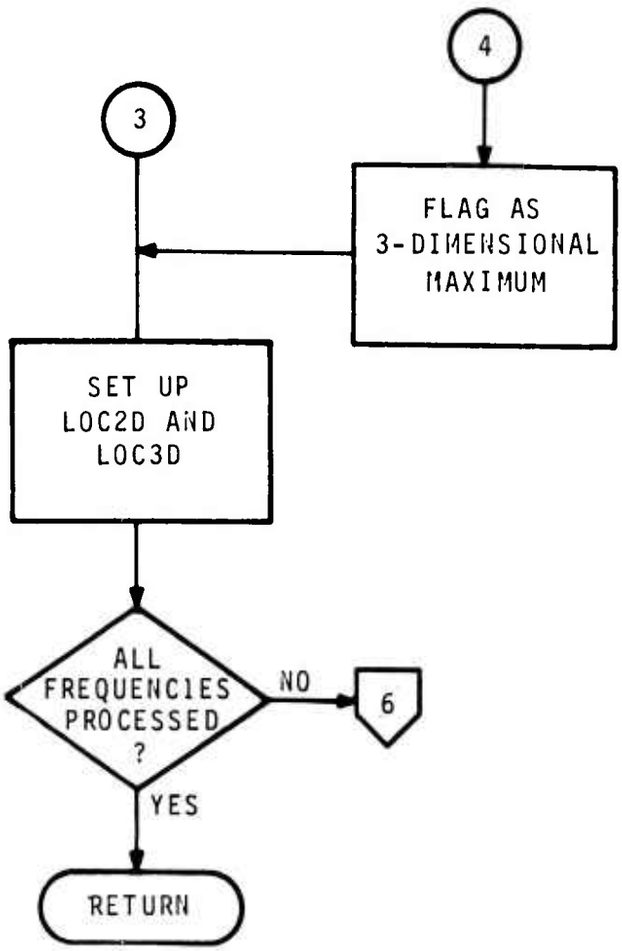
LOCATE(*) - PE INTEGER - location of point.

LOC2D(*,25) - PE INTEGER - index indicating frequency in which each 3-D maximum was found.

LOC3D(*,25) - PE INTEGER - index indicating frequency in which each 2-D maximum was found.


CHECKR
FPMAX(I)> FPMAX(I+1)
AND
FPMAX(I-1)
YES
4
6
NO
FPMAX(I-1)
>FPMAX(I)
?
YES
NO
FPMAX(I+1)
> FPMAX(I)
?
YES
NO
CALCULATE
WAVENUMBER
SEPARATION OF
FPMAX(I-1)
CALCULATE
WAVENUMBER
SEPARATION
OF
OF
FPMAX(I+1)
WAVENUMBER
SEPARATION
< DKX
?
YES
5
NO
WAVENUMBER
SEPARATION
< DKX
?
YES
5
NO
2


2
CALL REALE
RETRIEVE REAL ARRAY RESPONSE
CALL IMG
RETRIEVE COMPLEX ARRAY RESPONSE
CALCULATE POWER IN ADJACENT FREQUENCIES AT SAME WAVENUMBER LOCATION
FPMAX(I) > CHECK(1) ?
NO
5
YES
FPMAX(I) > CHECK(2) ?
YES
4
NO
5
FLAG AS 2-DIMENSIONAL MAXIMUM
3


4
3
FLAG AS
3-DIMENSIONAL
MAXIMUM
SET UP
LOC2D AND
LOC3D
ALL
FREQUENCIES
PROCESSED
?
NO
6
YES
RETURN

CNVTIM

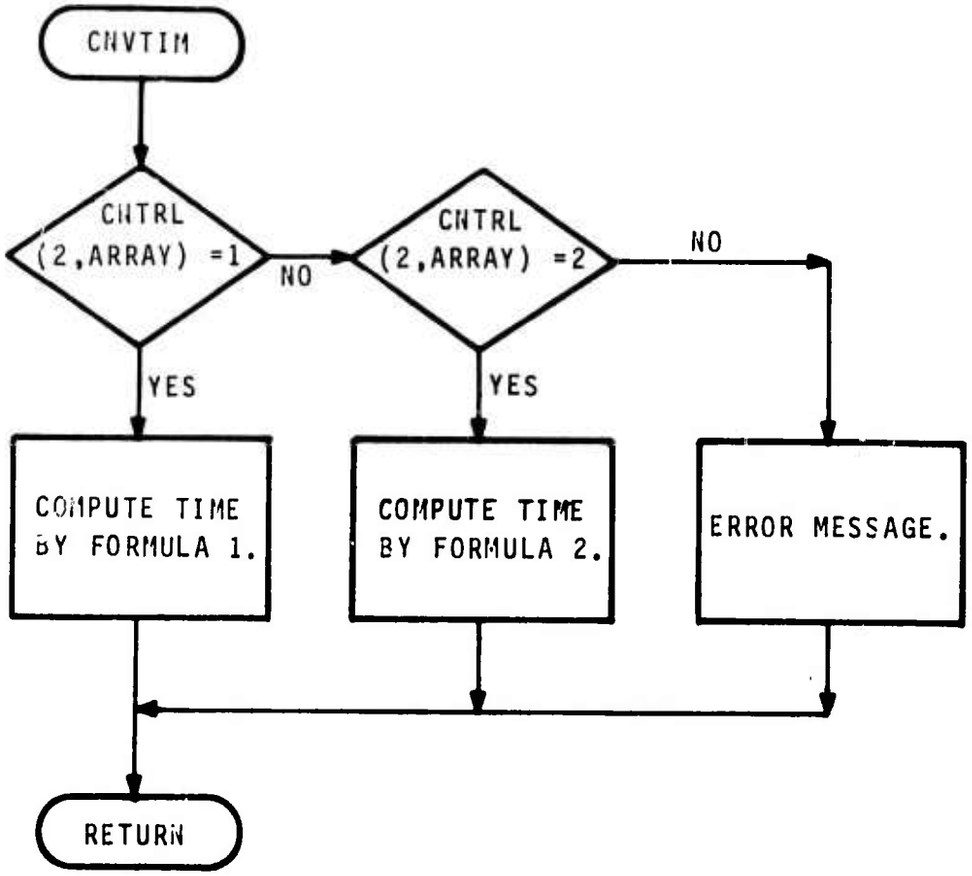
DESCRIPTION

CNVTIM is called when the next item to be retrieved is the timing data. The input pointer is pointing at the first byte of the timing data. The format of the timing data is dependent upon the site. Word 2 of CNTRL(*, ARRAY) give the format. Currently there are two formats accepted. If CNTRL (2,ARRAY) is equal to 1, the time is in the form of deciseconds from the beginning of the year and takes up 2 bytes. No conversion is necessary. The byte is retrieved and stored in TIME. If CNTRL(2,ARRAY) equals 2, the timing data is in 4 bit characters giving DDDHHMMSS. Three 16-bit bytes are retrieved and TIME calculated to be in deciseconds from the beginning of the year. If CNTRL(2,ARRAY) has a value other than 1 or 2, an error message is generated.

ARGUMENTS

CNTRL(2,ARRAY) - PE INTEGER - value indicates format of timing data.

TIME(*) - PE INTEGER - time in deciseconds from the beginning of the year as calculated by CNVTIM.


CNVTIM
CNTRL (2,ARRAY) =1
NO
CNTRL (2,ARRAY) =2
NO
YES
YES
COMPUTE TIME BY FORMULA 1.
COMPUTE TIME BY FORMULA 2.
ERROR MESSAGE.
RETURN

FNGRID

DESCRIPTION

FNGRID refines the value of the power maximum found on the coarse grid. It computes power in the east, south west and north directions about the point found on the coarse grid until the power does not increase in any direction.

A square grid is imposed around the point with a spacing between points which is one sixth (1/6) the spacing on the coarse grid. Figure 6 shows the square grid about p. the max on the coarse grid. The grid is refined the number of times specified by the user.

ARGUMENTS

REFINE - CU INTEGER - number of times to refine grid.
DKX - CU REAL - rectangular grid spacing.
KXMAX(*) - PE REAL - X-coordinate in K space of current power maximum.
KYMAX(*) - PE REAL - Y-coordinate in K space of current power maximum.
FPMAX(*) - PE REAL - power maximum for frequency being processed.

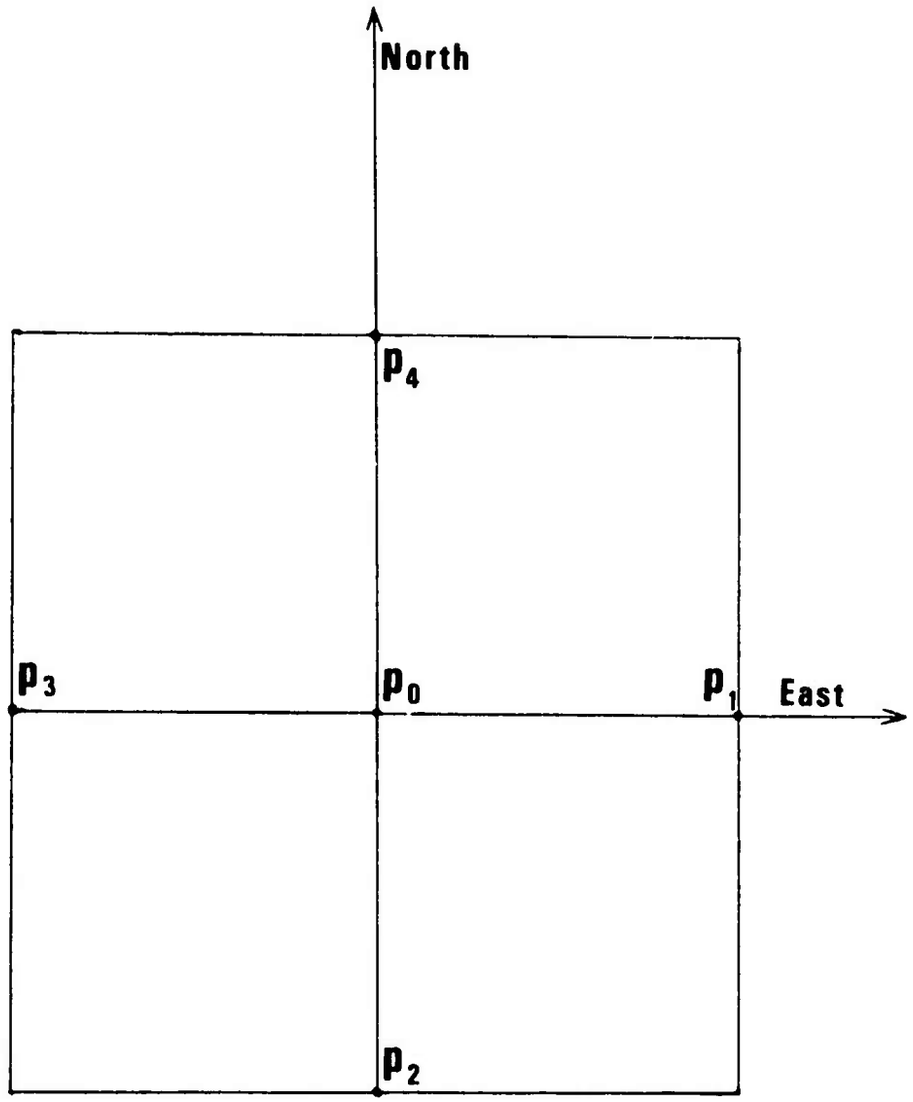
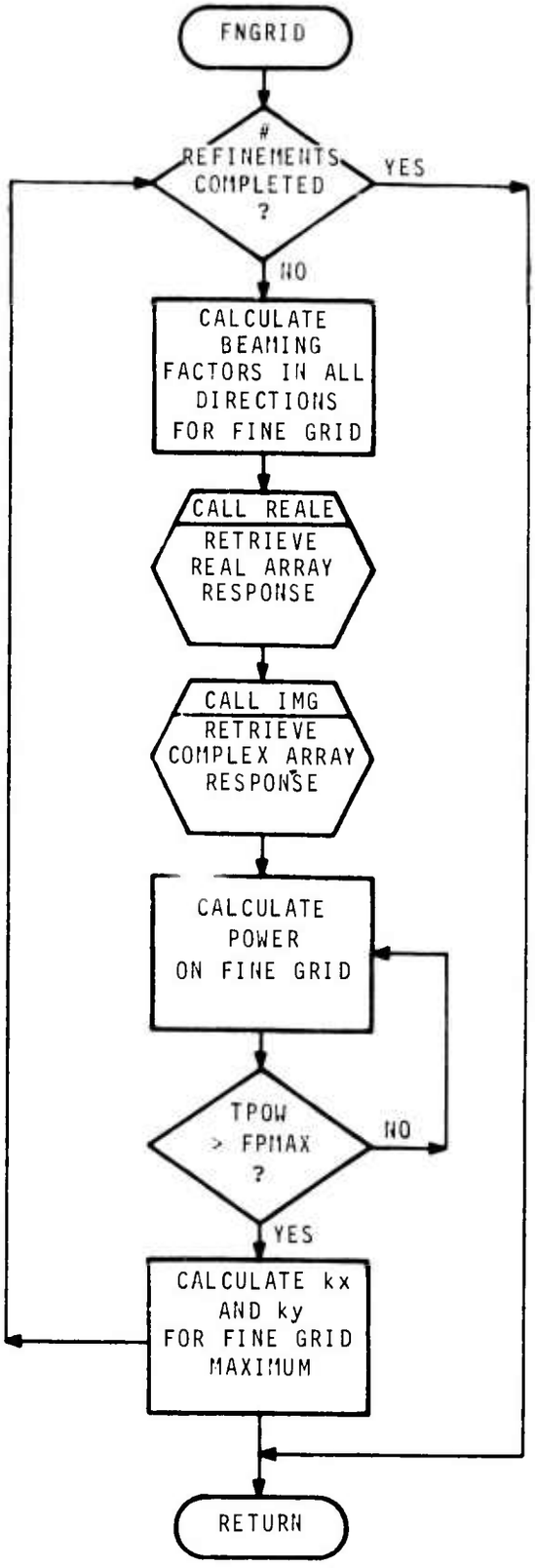


Figure 6. Diagrammatic Representation of Fine Grid.


FNGRID
REFINEMENTS COMPLETED ?
YES
NO
CALCULATE BEAMING FACTORS IN ALL DIRECTIONS FOR FINE GRID
CALL REALE
RETRIEVE REAL ARRAY RESPONSE
CALL IMG
RETRIEVE COMPLEX ARRAY RESPONSE
CALCULATE POWER ON FINE GRID
TPOW > FPMAX ?
NO
YES
CALCULATE kx AND ky FOR FINE GRID MAXIMUM
RETURN

GETBYT

DESCRIPTION

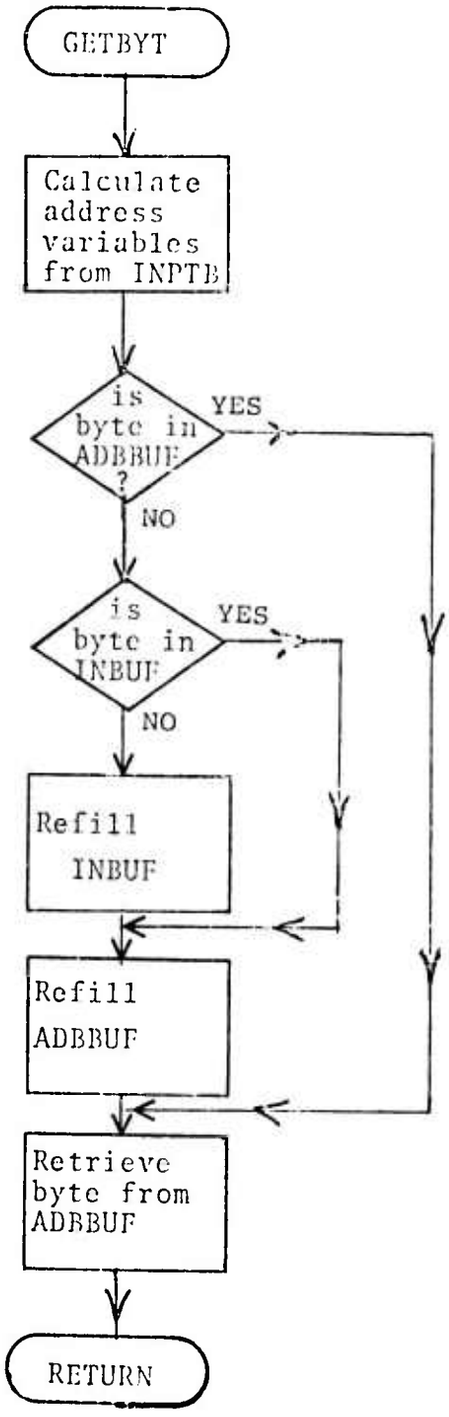
GETBYT retrieves the 16 bit byte pointed at by INBYT from the disk area INPUT. Two levels of buffering are used, ADBBUF in ADB and INBUF in core. First, a check is made to see if the address wanted is currently in ADB. If so, the byte is retrieved. Otherwise, the ADB buffer must be refilled from core. A similar check is made to see if the address is in core. If so, ADB is refilled and the byte retrieved. Otherwise, core is refilled from the appropriate location on disk, ADB refilled, and the byte retrieved. Initialization is such that on the first call, all buffers are marked empty and will be filled before the byte is retrieved. As long as bytes requested are consecutive, the refill of buffers will be minimal. Most bytes actually retrieved by DEM1 are consecutive, with the extra logic necessary to handle skips between time scans. Accesses are assumed to be always increasing, that is the address requested is assumed to be greater than the previous one.

ARGUMENTS

INPTB - CU INTEGER - serial number or address within disk area of byte desired.

INBYT - CU INTEGER - returns byte address by INPTB.

Flowchart


GETBYT
Calculate address variables from INPTR
is byte in ADBBUF ?
YES
NO
is byte in INBUF
YES
NO
Refill INBUF
Refill ADBBUF
Retrieve byte from ADBBUF
RETURN

GRID

DESCRIPTION

Grid computes the radius of the search disk for every frequency and the spacing between grid points in the X and Y direction. It passes the KX and KY coordinate for the first point on the grid and the change in X and Y directions to move progressively along the grid. On the second call to grid the actual KX and KY coordinates for the maximum power location found on the coarse grid are returned.

As mentioned above, a coarse triangular wavenumber grid is chosen initially in order to keep computing time to a minimum. However, the spacing must be fine enough to ensure that at least one point falls on a main lobe above the height of the largest side lobe of the array response. For example, if the largest side lobe of the array response has a peak 9 dB down from the maximum of the response, the grid spacing has to be selected so that at least one point falls within the 9 dB contour of the main lobe for any signal in the velocity range of interest.

In the initial coarse-grid computation of the wavenumber spectrum, FKCOMB employs an equilateral triangular grid because it requires fewer points and less computation time than does a square grid. Nevertheless, the convention has been adopted that users will specify the optimum square grid interval required for the given sensor array and FKCOMB will convert that to the corresponding equilateral triangular grid spacing (Figure 7). Figure 8 shows the point arrangement on the coarse grid.

ARGUMENTS

BORDER - CU REAL - width of border of search disk.

DKX - CU REAL - rectangular grid spacing.

ANGLE - CU REAL - radius of search disk.

LOWER - CU REAL - lower limit of phase velocity.

UPPER - CU REAL - upper limit of phase velocity

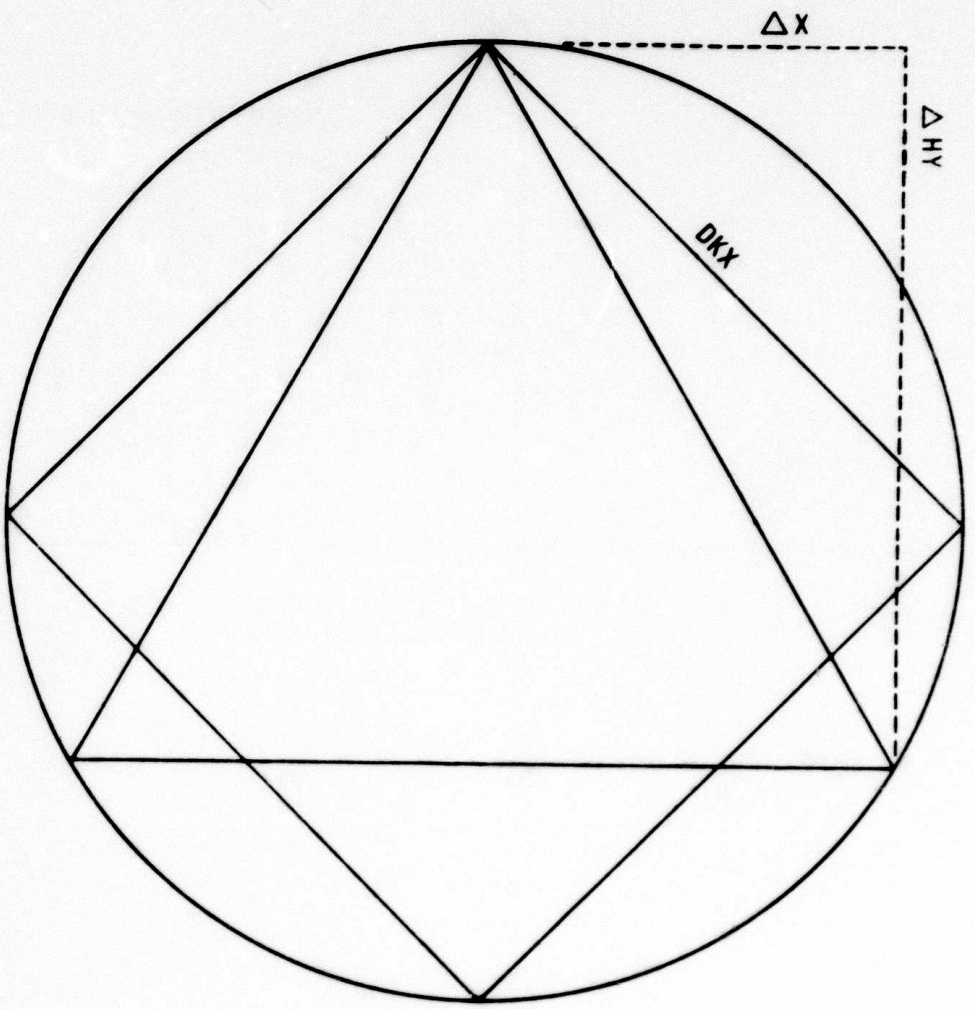
DELTX(500) - PE REAL - grid spacing on Y axis

DELTY(500) - PE REAL - grid spacing on X axis

KXMAX(*) - PE REAL - X coordinate in K space of current power maximum.

KYMAX(*) - PE REAL - Y coordinate in K space of current power maximum.

Coarse Grid Spacing



DKX is the grid-spacing input parameter calculated from the array response

ΔX is the spacing between points on the coarse grid in X direction

ΔY is the spacing between points on the coarse grid in the Y direction.

Figure 7. Coarse Grid Spacing.

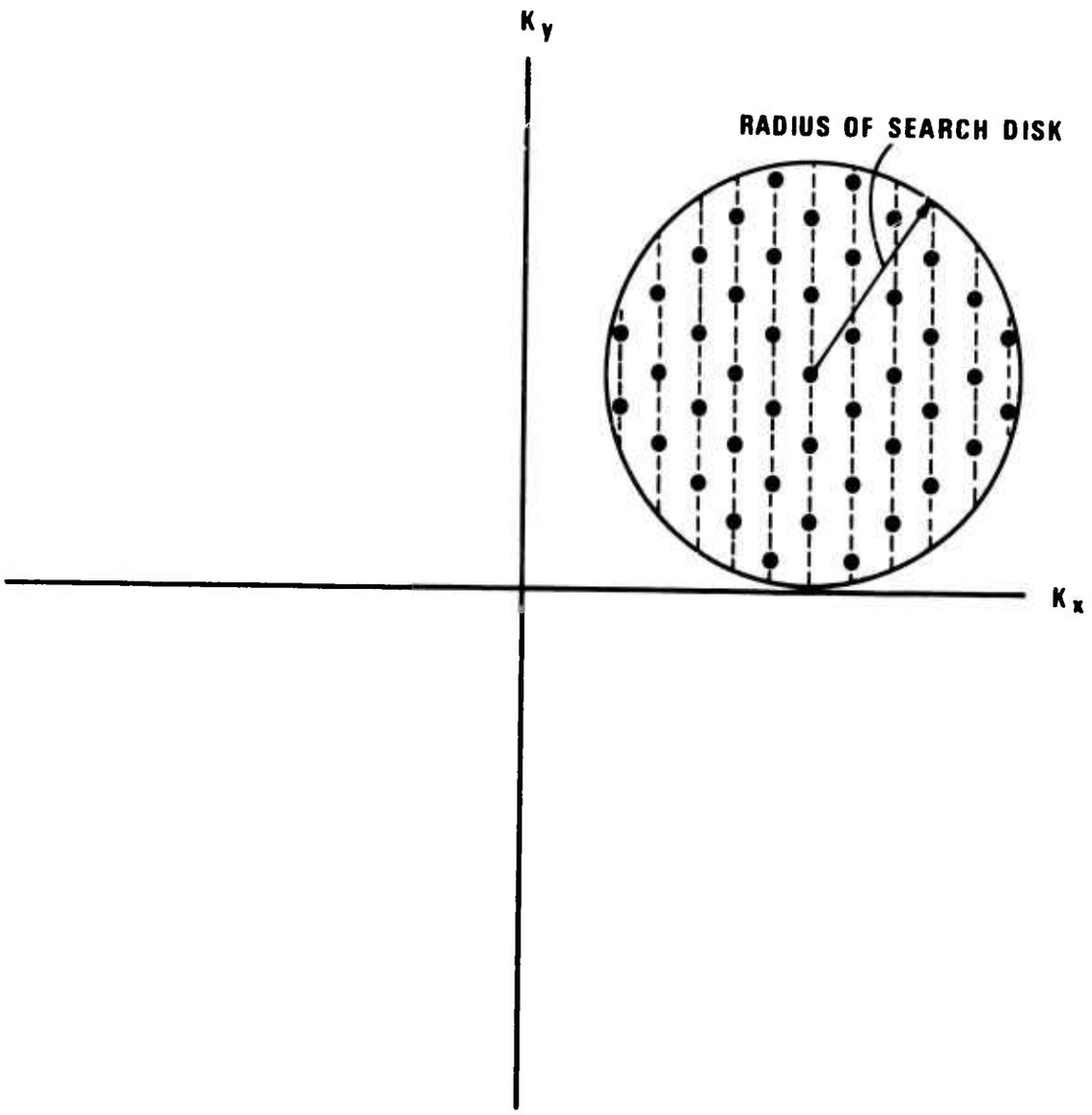
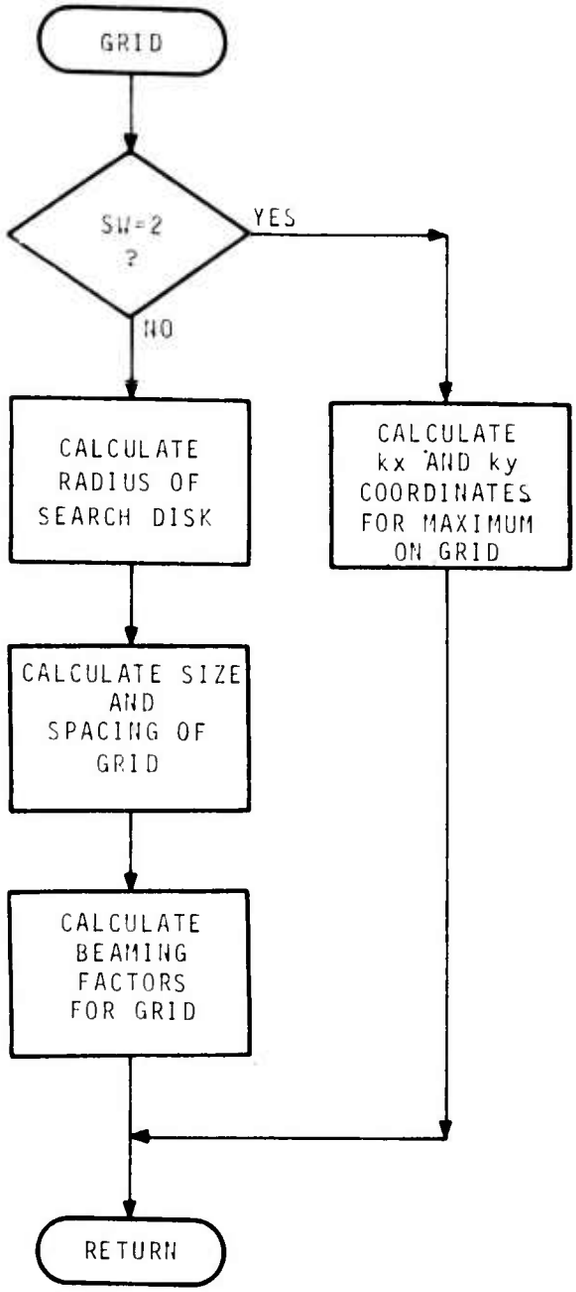


Figure 8. Point Arrangement on Coarse Grid.


GRID
SW=2
?
YES
NO
CALCULATE
RADIUS OF
SEARCH DISK
CALCULATE
kx AND ky
COORDINATES
FOR MAXIMUM
ON GRID
CALCULATE SIZE
AND
SPACING OF
GRID
CALCULATE
BEAMING
FACTORS
FOR GRID
RETURN

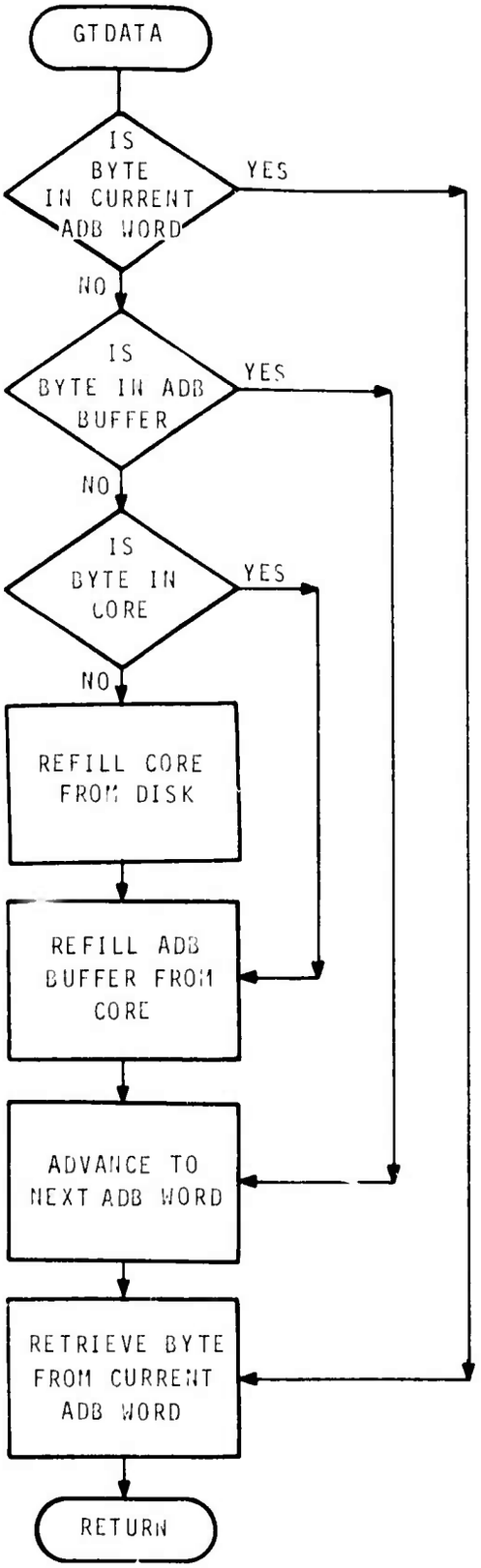
GTDATA

DESCRIPTION

GTDATA retrieves bytes sequentially from the disk area INDM2 (IN DEM2). The input data goes through two levels of buffering, one in ADB, one in core. Each call on GTDATA results in a 16-bit byte being placed, right justified, in INBYT, and the updating of pointers to the next 16-bit byte. The pointers are initialized in the main driver, DEM2, to reflect the fact that all buffers are empty before the first call.

ARGUMENTS

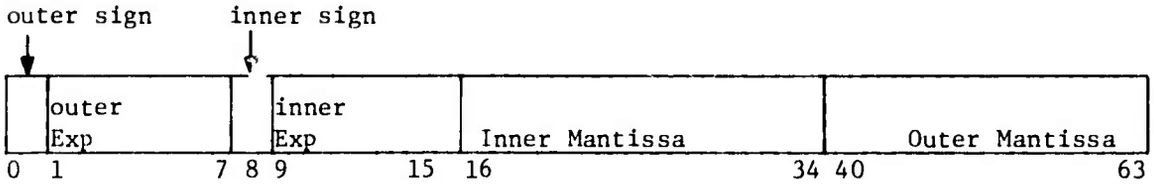
INBYT - CU INTEGER - returns the next 16-bit byte right justified and padded with zeroes.


GTDATA
IS BYTE IN CURRENT ADB WORD
YES
NO
IS BYTE IN ADB BUFFER
YES
NO
IS BYTE IN CORE
YES
NO
REFILL CORE FROM DISK
REFILL ADB BUFFER FROM CORE
ADVANCE TO NEXT ADB WORD
RETRIEVE BYTE FROM CURRENT ADB WORD
RETURN

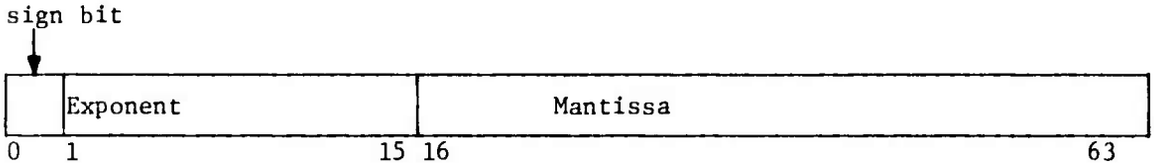
DESCRIPTION

IMG takes a complex number composed of a real and an imaginary part stored in the outer and inner 32-bit portions of a 64-bit wood, extracts the imaginary part, which is the inner part in ILLIAC terminology, and converts it to ILLIAC 64-bit floating point format. The routine consists almost entirely of assembly language code inserted in a CFD driver.

The complex number is in the following format:



The result is in the following format:



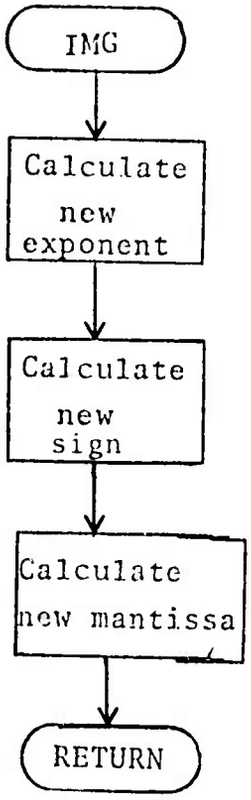
The sign bit is transferred without alteration. The mantissa is expanded from 24-bits to 48-bits by padding 24-bits of zero on the right. The exponent is calculated via the following formula:

NE = new exponent.
E = old exponent.
NE = (E-64)+16384.

ARGUMENTS

IN(*) - PE REAL - input complex number.
OUT(*) - PE REAL - output 64-bit floating point.

Flow Chart


IMG
Calculate new exponent
Calculate new sign
Calculate new mantissa
RETURN

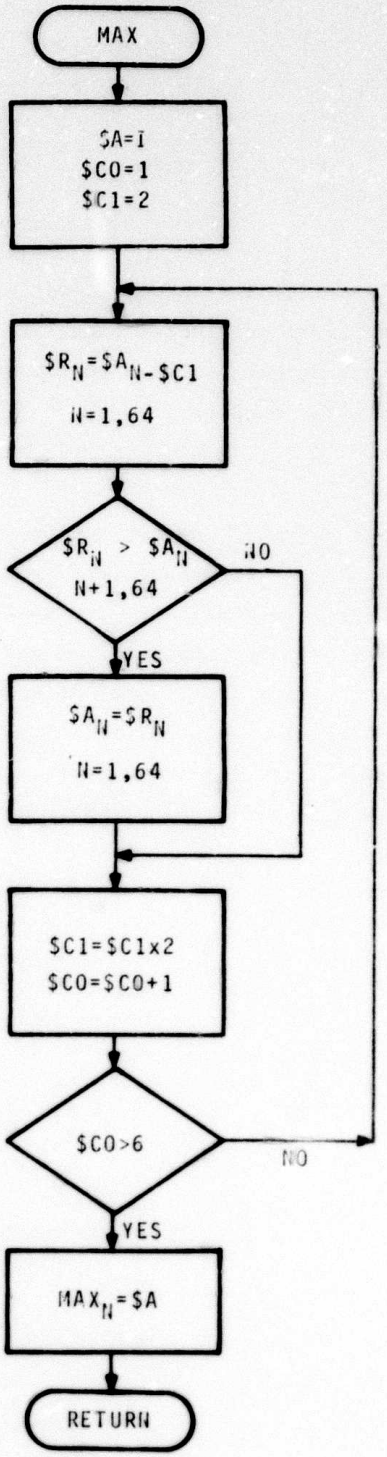
MAX

DESCRIPTION

MAX finds the maximum of 64 fixed point numbers one per PE, and returns that value in each PE. The method used is to compare each PE's value to that 1, 2, 4, 8, 16, and 32 away via the route instruction and to substitute the greater value each time. The mode is assumed to be ON upon entry. The inner branch is accomplished by temporarily inhibiting selective PE's. Subroutine MAX is used to determine the maximum value of a variable in each PE when initializing loops in the code. For example if a loop control is needed for the number of channels MAX determines the maximum number of channels stored in the PE's. The MODE is disabled in a given PE when the number of channels stored in it is reached. The MODE will be disabled in all PE's when the loop control equals the maximum number of channels found by subroutine MAX.

ARGUMENTS

I(*) - PE INTEGER - input parameter whose maximum is desired.
MAX(*) - PE INTEGER - maximum value of I(*). Identical in all PE's.


MAX
$A=I
$C0=1
$C1=2
$R_N=$A_N-$C1
N=1,64
$R_N > $A_N
N+1,64
NO
YES
$A_N=$R_N
N=1,64
$C1=$C1x2
$C0=$C0+1
$C0>6
NO
YES
MAX_N=$A
RETURN

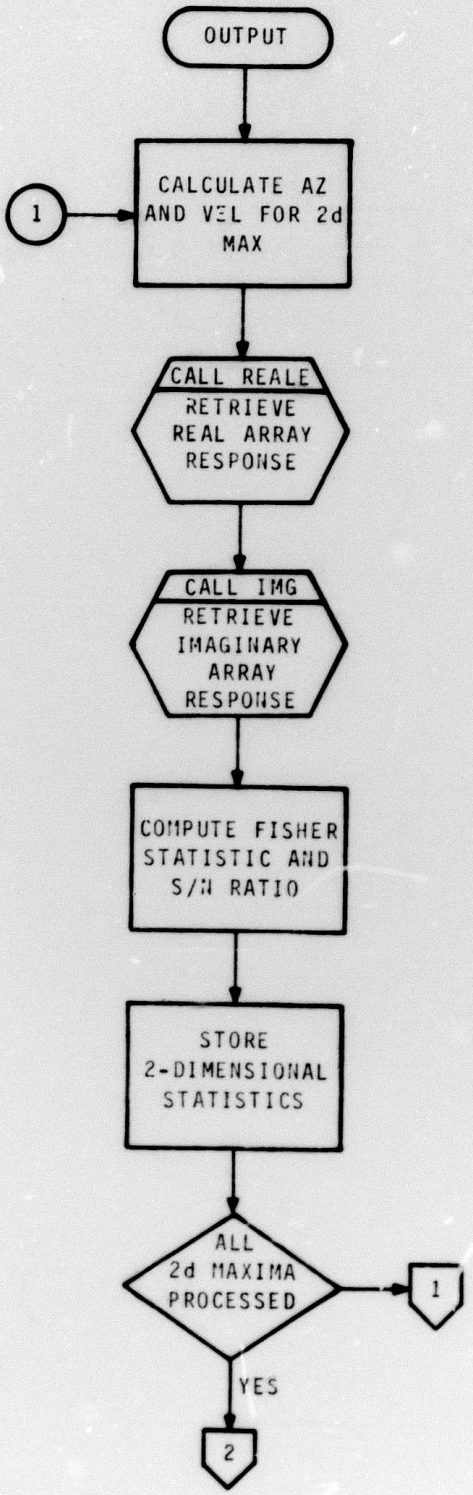
OUTPUT

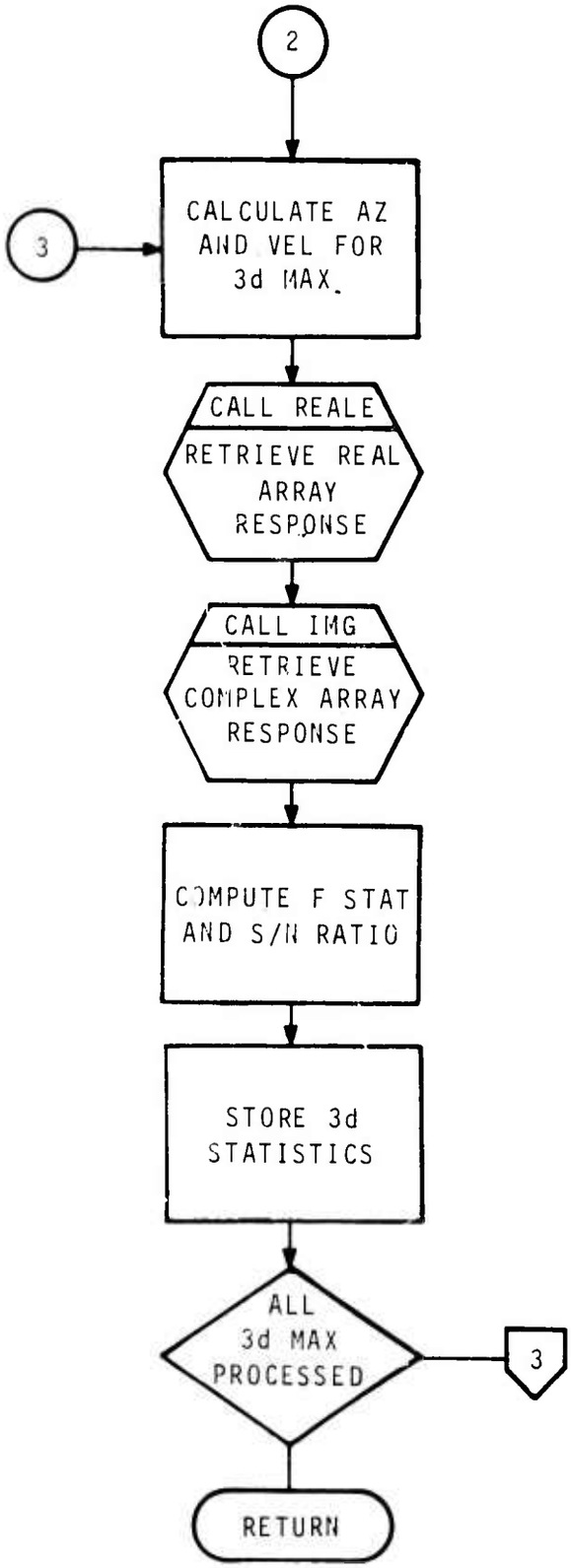
DESCRIPTION

The statistics associated with each power maximum are computed in subroutine output. The fisher statistic, signal to noise ratio, azimuth, velocity and period are output for each maximum all stored for each the time window. See flowchart for output.

ARGUMENTS

FPMAX(*) - PE REAL - power maximum for frequency being processed.
KXMAX(*) - PE REAL - X-coordinate in K space of current power maximum.
KYMAX(*) - PE REAL - Y-coordinate in K space of current power maximum.
AZ(*) - PE REAL - velocity of signal at maximum power.
VEL(*) - PE REAL - azimuth of signal at maximum power.
FSTAT(*) - PE REAL - fisher statistic at maximum power.
SIGNAL(*) - PE REAL - signal-to-noise ratio at maximum power.
PERIOD(*) - PE REAL - period of signal at power maximum.


OUTPUT
1
CALCULATE AZ AND VEL FOR 2d MAX
CALL REALE
RETRIEVE REAL ARRAY RESPONSE
CALL IMG
RETRIEVE IMAGINARY ARRAY RESPONSE
COMPUTE FISHER STATISTIC AND S/N RATIO
STORE 2-DIMENSIONAL STATISTICS
ALL 2d MAXIMA PROCESSED
1
YES
2


2
3
CALCULATE AZ AND VEL FOR 3d MAX.
CALL REALE
RETRIEVE REAL ARRAY RESPONSE
CALL IMG
RETRIEVE COMPLEX ARRAY RESPONSE
COMPUTE F STAT AND S/N RATIO
STORE 3d STATISTICS
ALL 3d MAX PROCESSED
3
RETURN

PUTBYT

DESCRIPTION

PUTBYT isolates the right hand 16 bits of OUBYT and inserts them in the output file indicated by ARRAY. The byte goes through two levels of buffering, a one word ADB buffer, ADBOUT and a 4096 word buffer (one for each output file) OUTBUF. Since there is only one ADB buffer, it is necessary to save and restore it whenever output arrays are changed. It is also necessary to empty the buffers when processing is completed. These functions are carried out by the main driver, DEM1.

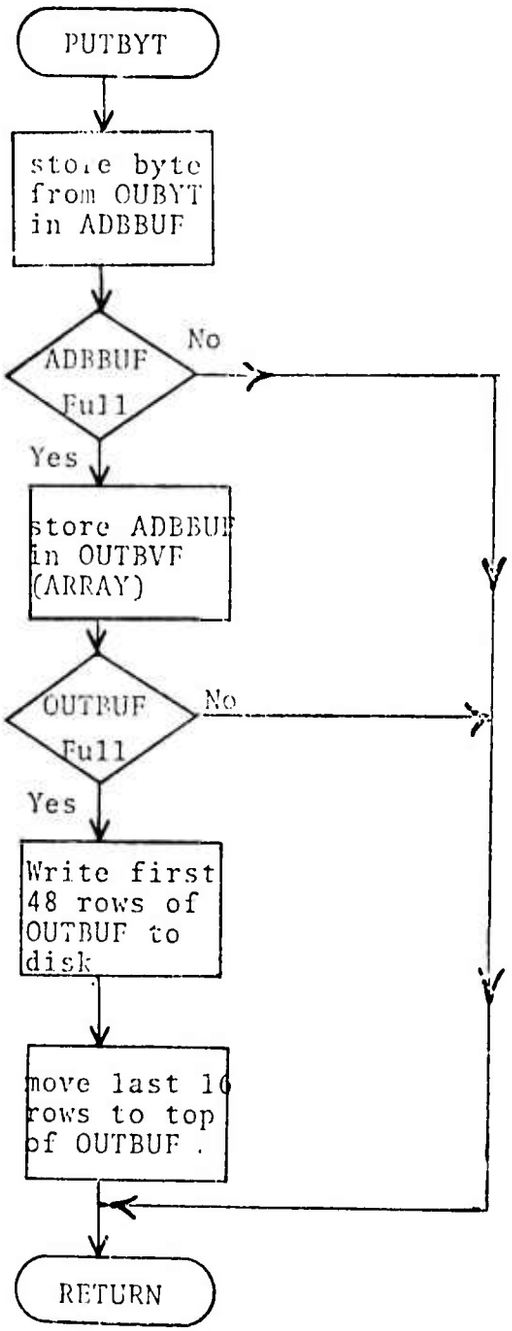
It is required that the most recent output always be available in core. For this reason, PUTBYT never empties the core buffers, but writes the first 75% to disk and moves the last 25% to the beginning of the buffer and positions the pointers appropriately.

ARGUMENTS

OUBYT - CU INTEGER - holds the byte to be output.

ARRAY - CU INTEGER - array being processed.

Flowchart


PUTBYT
store byte from OUBYT in ADBBUF
ADBBUF Full
No
Yes
store ADBBUF in OUTBVF (ARRAY)
OUTBUF Full
No
Yes
Write first 48 rows of OUTBUF to disk
move last 16 rows to top of OUTBUF .
RETURN

RDPRM

DESCRIPTION

RDPRM is provided to input by parameters necessary for DEM1 processing. Currently, the only parameter necessary is DEBUG, which controls debug printout. Since this is the only parameter, it is currently compiled in and no input from disk is necessary. Future expansion may require the addition of code to facilitate reading parameters from disk.

ARGUMENTS

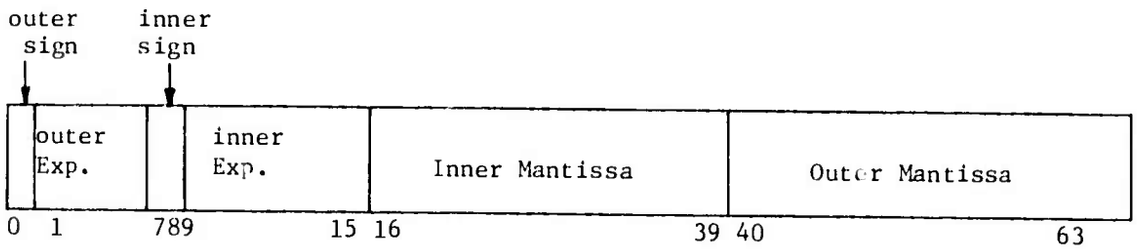
DEBUG - CU INTEGER - set to zero for routine processing.

REALE

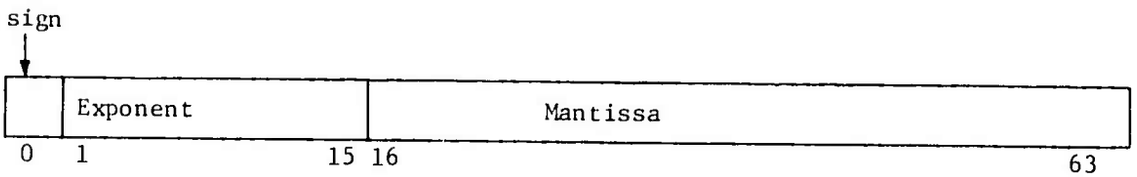
DESCRIPTION

REALE takes a complex number composed of a real and an imaginary part stored in the inner and outer 32-bit portions of a 64-bit word, extracts the real part, which is the outer part in ILLIAC terminology, and converts it to ILLIAC 64-bit floating point format. The routine consists almost completely of assembly language code inserted in a CFD driver.

The complex number is in the following format:



The result is in the following format:



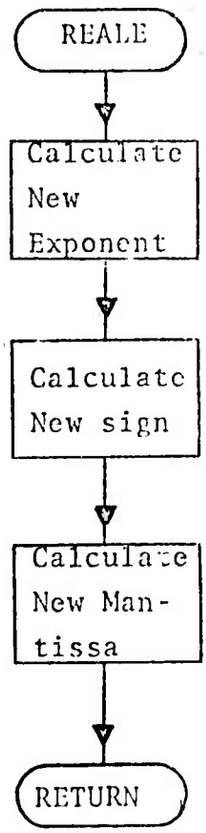
The sign bit is transferred without alteration. The mantissa is expanded from 24 bits to 48 bits by padding 24 bits of zero on the right. The exponent is calculated via the following formula:

NE = new exponent.
E = old exponent.
NE = (E-64)+16384

ARGUMENTS

IN(*) - PE REAL - input complex number.
OUT(*) - PE REAL - output 64-bit floating point.

Flowchart


REALE
Calculate New Exponent
Calculate New sign
Calculate New Man-tissa
RETURN

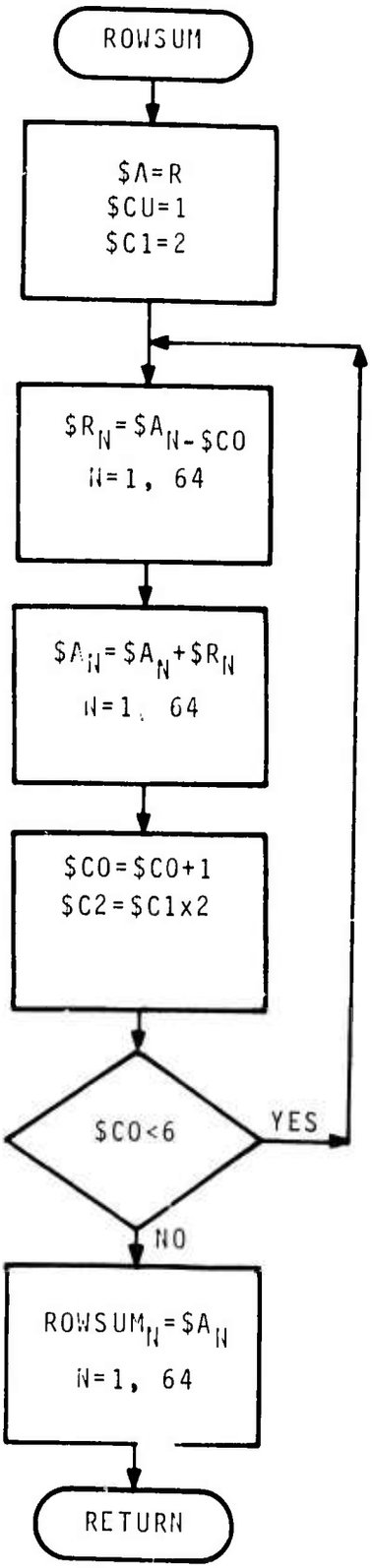
ROWSUM

DESCRIPTION

Function ROWSUM returns the sum of 64 floating point numbers, one per PE. The result is identical in all PE's except for differences caused by adding the numbers in different orders. The algorithm used is the "standard ILL AC logsum" method whereby partial sums are compiled for each sequence of 2, 4, 8, 16, 32, and then all 64 PE's by routing by 1, 2, 4, 8, 16, and 32. The routine is coded via insered ASK statements within a CFD driver. It is assumed that MODE=ON when ROWSUM is called.

ARGUMENTS

R(*) - PE REAL - the numbers to be summed.
ROWSUM(*) - PE REAL - the result, replicated in all PE's.


ROWSUM
$A=R
$CU=1
$C1=2
$R_N=$A_N-$CO
N=1, 64
$A_N=$A_N+$R_N
N=1, 64
$CO=$CO+1
$C2=$C1x2
$CO<6
YES
NO
ROWSUM_N=$A_N
N=1, 64
RETURN

RUNFFT

DESCRIPTION

RUNFFT is a CFD driver which makes the University of Illinois FFT routine accessible to a CFD program as a normal subroutine call. It is primarily assembly language code to facilitate the difference in parameter linkage. The FFT routine is called with the following parameters:

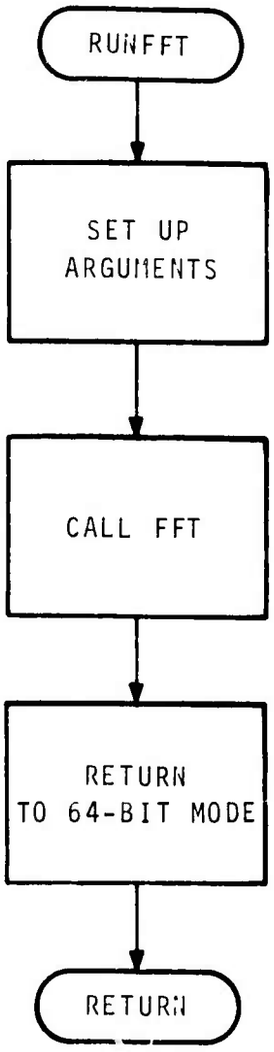
time window size - TWSZ
number of time windows - NGDCH
input data - BUFF2(*,1,OLD)

The FFT routine requires that the parameters be pointed to by acar 2. Acar 2 contains the address of a 3 word block. Each word of this block contains the address of one of the arguments. Word 2 give the number of discrete time windows. The third word contains the address of the data to be FFT'ed. The University of Illinois routine currently leaves the machine in 32-bit mode. The driver returns the machine to 64-bit mode before processing continues.

ARGUMENTS

The arguments to FFT are passed in common via ADB. The variables involved are:

TWSZ - CU INTEGER - power of 2 between 64 and 512 specifying time window size.
NGDCH - CU INTEGER - number of time windows, to FFT.
BUFF2(*,1,OLD) - PE REAL - passes input data and output data.


RUNFFT
SET UP
ARGUMENTS
CALL FFT
RETURN
TO 64-BIT MODE
RETURN

OPERATIONAL PROCEDURES

DATA PREPARATION

Two input files must be supplied by the user at run time. The first, RUN.INPUT, contains the raw input as read from a long period tape. This data is currently provided by reading the tape on the 360/44 at SDAC and shipping the resulting data file to the ILLIAC site over the ARPANET via FTP.

The second file, RUN.PARAM, contains run-time parameters in binary form. There is currently no straight forward means for creating such a file. During testing of the program, the file was created by assembling and running a job on ILLIAC which writes the data to a disk file. The parameters are assembled in so that the assembler converts the data from symbolic form to binary form. The source of the assembly language program currently used is:

```
BEGIN
START::
PUT CONPRM,CONPRMMEM,COMPLWD:
PAUSE COMPLWD;
HALT;
HALT;
CONPRM:AREA "CONPRM";
COMPLWD:WDS 1;
BLK;
CONPREMMEM:DATA
0,              % DEBUG=0
64,             % TIME WINDOW LENGTH=64
0,              % OVERLAP=0
10.0,           % GLITCH FACTOR=10.0
10.0            % VARIANCE FACTOR=10.0
2,              % FREQUENCY 2 IS LOWEST FREQUENCY
9,              % FREQUENCY 9 IS HIGHEST FREQUENCY
0,              % 0 MEANS VERTICAL MOTION
0,              % THIS WORD IS NOT USED
0,              % THIS WORD IS NOT USED
```

```
0,              % THIS WORD IS NOT USED
.004,           % DKX
2.90,           % LOWER VELOCITY LIMIT.
9999999.,       % UPPER VELOCITY LIMIT
0.0,            % ANGLE=0.0; SEARCH IN ALL DIRECTIONS.
2,              % REFINE FINE GRID TWICE
1,              % SAMPLING RATE IS ONE PER SECOND
0;              % END OF PARAMETERS.
END START.
```

The primary input file used to run this job, assuming that the above source is contained in the file PARAM.ASK is:

```
ASK PARAM.ASK,PARAM.REL,PARAM.LIST
LINKED
ISV PARAM.ISV
INCLUDE PARAM.REL
END
ALLOC FKCOMB.MAPFILE, 1
RUN PARAM.ISV
MOVE 14DM:CONPRM, RUN.PARAM
DALLOC 1
```

PROGRAM EXECUTION

In order to run the ILLIAC Long-period Seismic Array Processing System the following files must be stored on the ILLIAC central file system:

DEM1.ISV
DEM2.ISV
FKCOMB.ISV
FKCOMB.MAPFILE
RUN.INPUT
RUN.COORDINATES
RUN.PIF
RUN.PARAMETERS.

All of the above files except RUN.PARAMETERS and RUN.INPUT are provided. RUN.PARAMETERS contains user specified run time parameters and should be prepared for each run. RUN.INPUT contains the long-period seismic data. See Data preparation, section Iv.b for description of these files.

RUN.PIF (primary input file) contains the job control language that follows:

```
RUN.PIF
ALLOC FKCOMB.MAPFILE,1
MOVE RUN.COORDINATES,14DM:STCORD
MOVE RUN.PARAM,14DM:CONPRM
MOVE RUN.INPUT,14DM:INPUT
RUN DEM1.ISV,MAXTIM=600
RUN DEM2.ISV,MAXTIM=600
RUN FKCOMB.ISV,MAXTIM=600
MOVE 14DM:FKDISP,RUN.OUTPUT
DALLOC 1.
```

The command ACL "SUBMIT RUN.PIF" enters the job onto the ILLIAC batch queue. For details of job submittal and monitoring see ACL USER's GUIDE. Upon completion the disk Files RUN.OUTPUT and RUN.POF will have been created. RUN.POF contains the system responses to the job command. RUN.OUTPUT is an 8-bit ASCII file suitable for printing containing bulletins for all 2-D and 3-D maximums found and the associated statistics.

GLOSSARY

ACAR - any one of four accumulators in the CU.

ADB - Advast Data Buffer. 64 words of fast access memory used by the CU.

ASK - refers to the ILLIAC assembler and the assembly language.

CFD - a Fortran like language developed by NASA which compiles on an IBM 360 and outputs ILLIAC assembly language.

Components of motion - the vertical, north, or east direction of wave motion.

CU - control unit. The ILLIAC Control Unit decodes the instruction stream to be executed by the PE's. It also executes a limited set of instructions.

Data record - a number od data scans. If not every data scan has a time word, the data record will contain a time word corresponding to the time of the first scan.

Data scan - a unit containing the values for all the data channels at a seismic station for a given instant, which may also be on the scan. During data retrieval one scan at a time is processed.

MODE - a pattern of 64-bits which controls the write protection of registers and memory for each PE.

PE - processing element. One of 64 ILLIAC processors which execute the same instruction stream in lock step.

Route instruction - provides an efficient means for communication between PE's.

Time word - word holding the time of the first data scan in a data frame. Because the data scans are spaced equally in time, the time of any scan in the frame is known if the time of the first scan is known. GMT is used in all time words.

REFERENCES

Blandford, R. R., 1971, An automated event detector at TFO: Seismic Data Laboratory Report No. 263, Teledyne Geotech, Alexandria, Virginia.

CFD, A Fortran Based Language for ILLIAC IV, 1973, Computational Fluid Dynamics Branch, Ames Research Center, National Aeronautics and Space Administration.

ILLIAC IV Systems Characteristics and Programming Manual, Burroughs Corporation Defense, Space and Special Systems Group. January, 1971.

Mack, H., 1972, Evaluation of the LASA, ALPA, NORSAR long period network: Seismic Array Analysis Center Report No. 6, Teledyne Geotech, Alexandria, Virginia.

Smart, E., 1971, Erroneous phase velocities from frequency wavenumber spectral sections: Geophys. Journ. Royal Astr. Soc., v. 26, Nos. 1-4, p. 247-254.

Smart, E. and Flinn, E. A., 1971, Fast frequency-wavenumber analysis and Fisher signal detection in real time infrasonic array data processing: Geophys. Journ. Royal Astr. Soc., v. 26, Nos. 1-4, P. 279-284.

Stevens, J. E., 1971, A Fast Fourier Transform Subroutine for ILLIAC IV: C.A.C. Document No. 17, Center for Advanced Computation, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801.

System Guide for the ILLIAC IV User, 1974, Institute for Advanced Computation, Ames Research Center, Mullett Field, California 94035.