5

AD A095932

DTNSRDC/CMLD-81/01

DAVID W. TAYLOR NAVAL SHIP
RESEARCH AND DEVELOPMENT CENTER

Bethesda, Maryland 20084

12 LEVEL II

FREQUENCY TRACK STORAGE FOR A SIGNAL ANALYSIS SYSTEM

by

Christopher J. Slominski

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED

COMPUTATION, MATHEMATICS AND LOGISTICS DEPARTMENT
DEPARTMENTAL REPORT

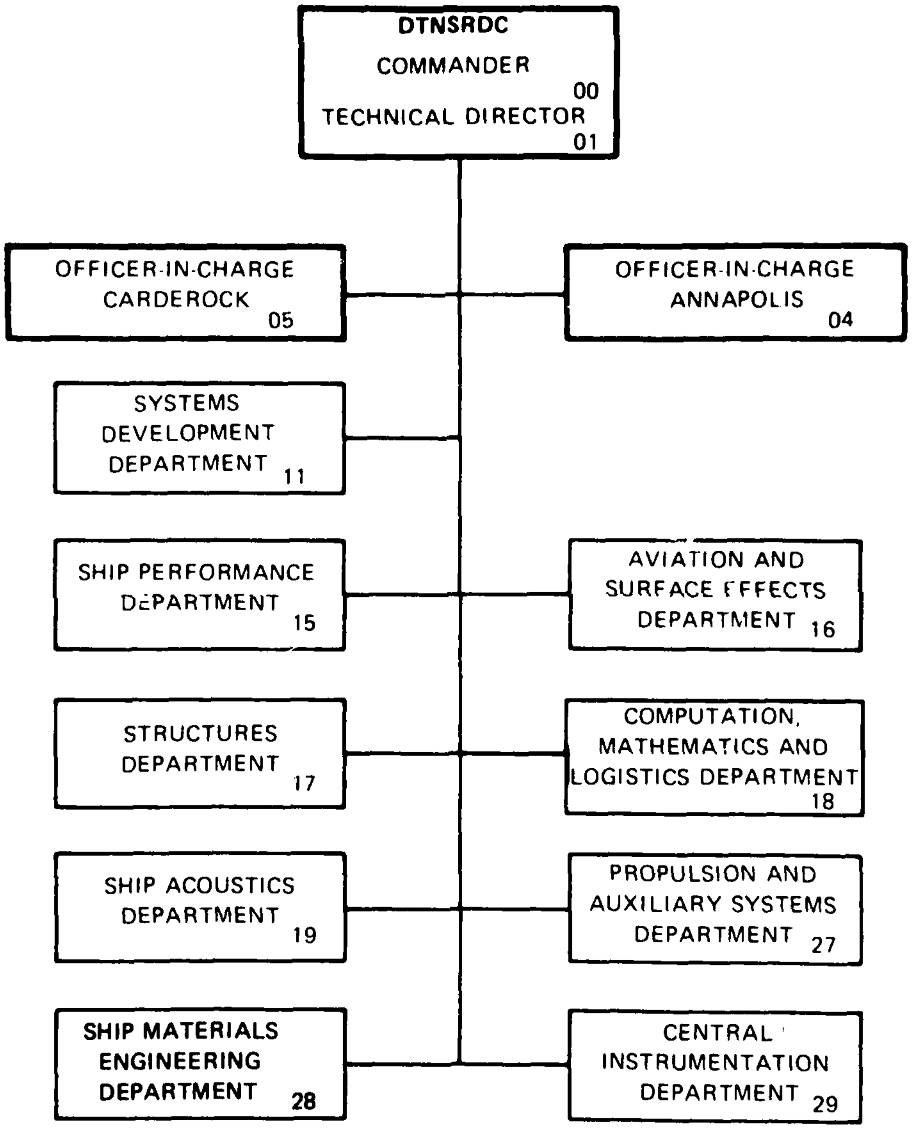
January 1981

DTNSRDC/CMLD-81/01

FREQUENCY TRACK STORAGE FOR A SIGNAL ANALYSIS SYSTEM

81 3 04 018

MAJOR DTNSRDC ORGANIZATIONAL COMPONENTS


DTNSRDC
COMMANDER 00
TECHNICAL DIRECTOR 01
OFFICER-IN-CHARGE CARDEROCK 05
OFFICER-IN-CHARGE ANNAPOLIS 04
SYSTEMS DEVELOPMENT DEPARTMENT 11
SHIP PERFORMANCE DEPARTMENT 15
AVIATION AND SURFACE EFFECTS DEPARTMENT 16
STRUCTURES DEPARTMENT 17
COMPUTATION, MATHEMATICS AND LOGISTICS DEPARTMENT 18
SHIP ACOUSTICS DEPARTMENT 19
PROPULSION AND AUXILIARY SYSTEMS DEPARTMENT 27
SHIP MATERIALS ENGINEERING DEPARTMENT 28
CENTRAL INSTRUMENTATION DEPARTMENT 29

UNCLASSIFIED

16 SR 01403

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

14

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER DTNSRDC/CMLD-81/01	2. GOVT ACCESSION NO. AD-A095032
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) 6 FREQUENCY TRACK STORAGE FOR A SIGNAL ANALYSIS SYSTEM	5. TYPE OF REPORT & PERIOD COVERED
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) 10 Christopher J. Slominski	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS David Taylor Naval Ship Research and Development Center Bethesda, Maryland 20084	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61153N, SR01403, 17 SR0140301 1808-010
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Sea Systems Command Code 03F Washington, D.C. 20360	12. REPORT DATE 11 Jan[redacted]81
	13. NUMBER OF PAGES 27
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 12 32	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)

Frequency spectrum
Signal Analysis
Graphics Data Base

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

The problem of Frequency track storage is one small part of implementing a comprehensive signal analysis system. This paper deals only with their storage and does not discuss the signal analysis problem. The frequency tracks are used in signal classification and an optimal data base allowing graphics and statistical analysis is sought.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

406847 [illegible]

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

The problem of Frequency track storage is one small part of implementing a comprehensive signal analysis system. This paper deals only with their storage and does not discuss the signal analysis problem. The frequency tracks are used in signal classification and an optimal data base allowing graphics and statistical analysis is sought.

INTRODUCTION

This paper deals with the topic of frequency track storage. The stored frequency tracks will be used in a graphics oriented signal analysis system requiring graphics data base storage. The requirements on the data base will be given followed by several possible schemes to accomplish the task. The best choice of the schemes will be made according to how well it satisfies the requirements.

Before the topic of frequency track storage is pursued, an explanation of what frequency tracks are and why they are useful will be given.

Continuous acoustic data is recorded as a voltage amplitude over time. The signal is then digitized and stored as discrete voltage amplitudes sequentially written onto disk. By using a number of points from the digitized wave form, a fast fourier transform (fft) produces the representation of the wave, over the given sample points, in the frequency domain. The output of the fft is a series of complex numbers defining discrete frequency bins between zero and half the sample rate of the original data. For example, sampling the continuous wave at a rate of 18,000 samples per second gives a fft output range of 0 to 9000 cycles per second. By squaring and adding the real and imaginary parts of each complex number along the frequency domain, the power spectrum for that set of data-points is computed. The power spectrum is the power used at the various frequencies to create the wave (see figure #1). The power peaks in the spectrum are the objects of interest for analysis of the acoustic data. Each peak has four parameters associated with it. These are amplitude, band width at the half power point, frequency at which the peak occurs, and phase. The phase of a peak is a parameter obtained from the original complex number. A vector is defined between the origin of the complex plane and the complex number for the power peak. The angle between the real axis and the vector is the phase of the power peak. The phase is computed as PHI=arctan (y/x) where y is the imaginary component and x the real component of the complex number. Taking power spectra for a number of points repeatedly, while moving along the data for each new power spectra, yields a related set of power spectra in time.

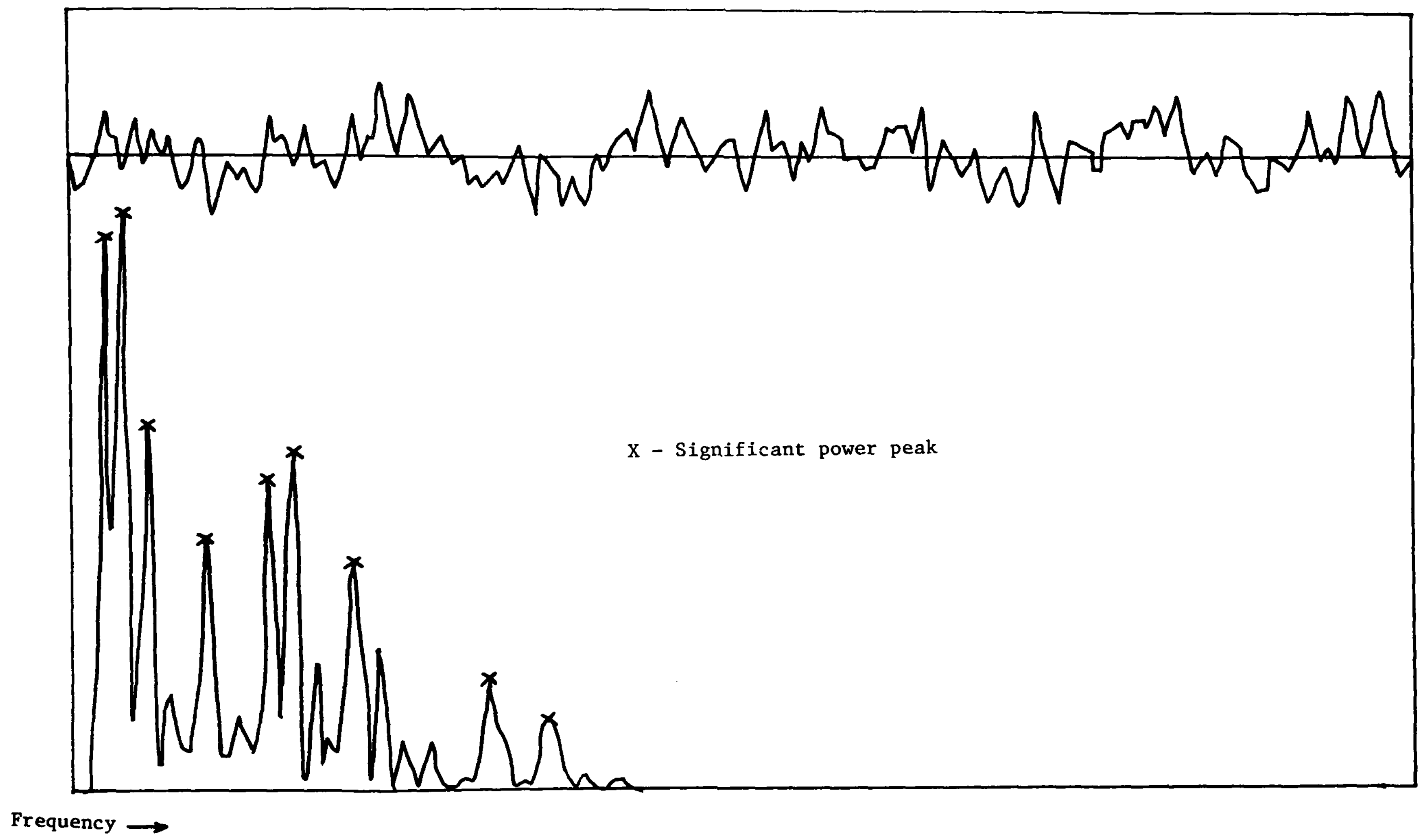


figure #1 "Signal and power spectrum"

The set of power spectra is used to observe the frequency distribution of power over time of a signal. The connecting of power peaks on different spectra (time slices) in the set is called tracking. The tracking algorithm decides which peaks belong to a track by frequency bandwidth and amplitude considerations (see figure #2). Figure #2 shows thirty power spectra (time slices) which have had the peaks connected to form tracks. Note the possibility of tracks splitting and merging. Also note that tracks can cross time slices where no peak is marked since the tracking algorithm fills in a peak when it is deemed necessary in order to continue a track.

The purpose for examining tracks is to extract unique characteristics about various signal phenomena. Once the ability to extract significant features from data is achieved, one can classify data into groups without knowledge of the signal's source.

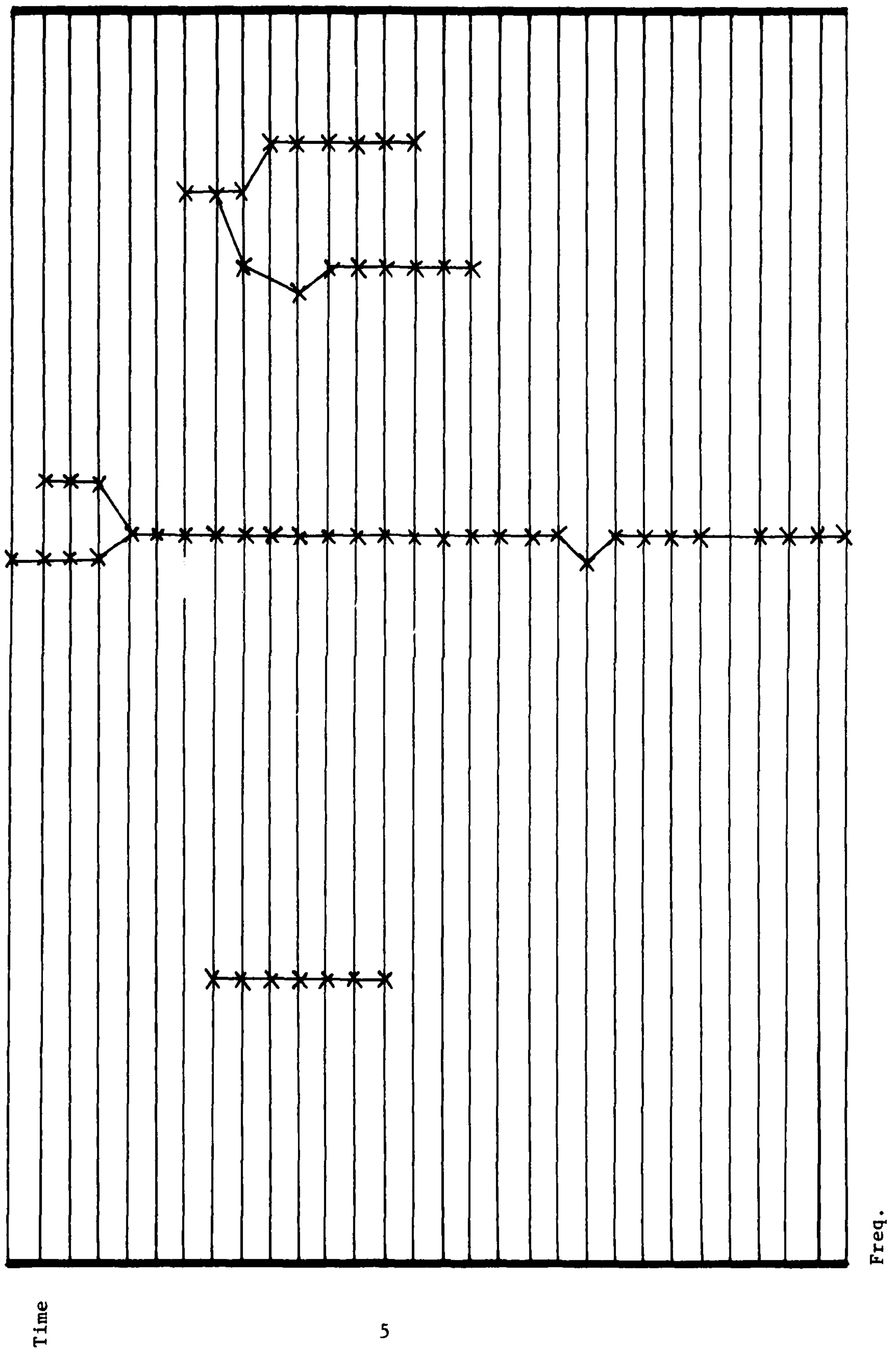


figure #2 "TRACKS"

OBJECTIVES

We begin by stating the requirements imposed upon the frequency track storage data base.

LOW STORAGE REQUIREMENTS

Given several viable alternative storage schemes, the one with the least storage requirements should be given greatest consideration. Low storage is especially important with regard to core memory storage. For disk storage, size is not as important as whether the data on disk can be partitioned into smaller sets that are independently useful and can be placed in core memory quickly.

EFFICIENT DISPLAY OF DATA

The tracks are to be displayed on a C.R.T. for the user's examination. Many power spectra usually are taken for a signal but for resolution purposes, not all time slices will be displayed on the screen at one time. Figure #2 earlier showed a sample display. The number of time slices that can be seen at one time with satisfactory resolution is set at 30. The displaying mechanism must be able to display time slices n thru n + 29 where n is any number between 1 and the total number of power spectra taken (minus 29). For resolution purposes the whole frequency range of the power spectra also is not displayed. Proper resolution is achieved when a frequency band for the display is set for n to n + 1799 Hz. Each display therefore is a window in time and frequency into the data. The ability to create the display implies sequential ordering of the peak data in time.

EFFICIENT EXAMINATION OF A DISPLAY

While a display window is being viewed, the ability to observe the underlying phase, bandwidth, and amplitude data is desired. A method that allows the user to move from peak to peak on the display and have it's 4 vector of information displayed is needed. The faster this can be done, the greater it will enhance the perception of the data being viewed.

PROPER HANDLING OF INTERSECTIONS

As was seen in figure #2, tracks can have intersections. If tracks will not be stored as complete units (branches stored seperately) the decision as to which branch will contain the junction peak must be made. Both branches may have to contain the peak, creating some redundancy.

STORAGE GROUPING

In the discussion of frequency track storage, one of the three objects shown in figure #3 will be used as a basic storage structure.

1. PEAK:

Each individual peak could be stored with indices to the 2 peaks it connects, both upward and downward in time.

2. SEGMENT:

Each connected set of peaks having no more than one peak per slice can be stored with indices referring to segments joining the top or bottom peaks.

3. TRACK:

The complete disjoint set of peaks can be stored as an independent unit.

These three objects form the basis for many variations of the general algorithms for storage that will be mentioned. All three objects will not always be applied in discussion of methods.

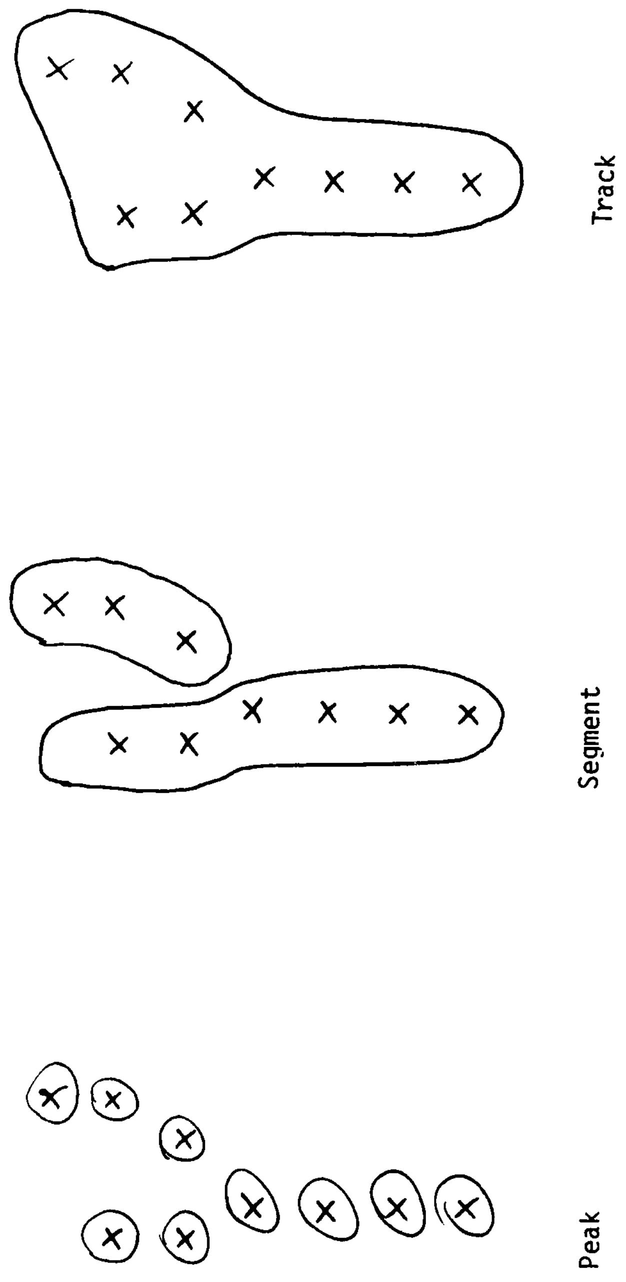


figure #3 "Storage grouping"

TRACK MATRIX

The first method to be considered will be called the "Track Matrix". As can be seen in figure #4, this is a set of fixed length arrays.

The vertical axis of figure #4 labeled "time" is the maximum number of power spectra that can be tracked. Each segment has this total number of time slices reserved with space for the 4-vector of information on each time slice. The track Matrix uses segment storage grouping since the array allows for only one peak per time slice per stored object. Every segment found is stored in its own segment array over the appropiate time slices. Time slices not occupied by that segment are zeroed out.

A standard example of 500 power spectra and 200 tracks (or segments) will be used for the track matrix and later methods to be considered. The standard will be matched against the objectives stated earlier for all methods.

Track Matrix storage requirements for the example are easily computed. Frequency, phase, and bandwidth numbers each require a word of storage. Amplitude data is stored as a real number requiring two words. The computation is as follows:

$$5\ \frac{\text{words}}{\text{peak}} \times 500\ \frac{\text{peaks}}{\text{segment}} \times 200 \text{ segments} = \underline{500{,}000 \text{ words}}$$

500,000 words is high compared to other methods. The reason for large storage requirements can be understood as follows. Most segments only run about 25 time slices on the average. The segment arrays therefore have many spaces wasted (zeroed out) because the segment is not the maximum length (500 slices).

Creating a display on the C.R.T. is easily accomplished because of the indexing ability of the arrays and the sequential time ordering of the segments. Any time slice can be found quickly and all segments containing an entry for that time slice can be noted.

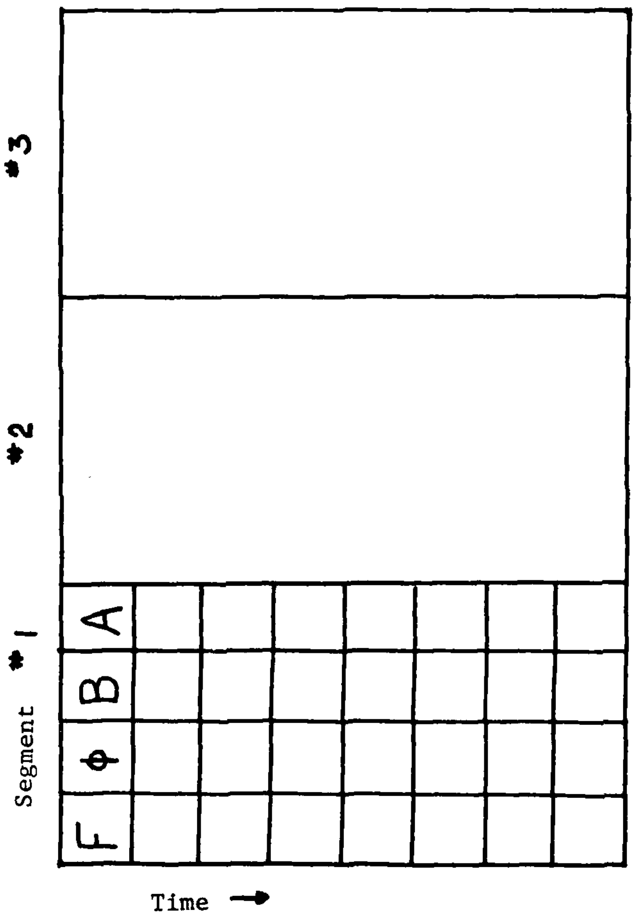


Figure 4 - Track Matrix

For the same reasons as above, examining the display is efficient. Those peaks on a segment pertinent to the display screen can be read into core memory quickly. Retrieval of the information from the peaks in core memory will be fast and easy.

The last objective to be considered is proper handling of intersections. The storage structure for the track matrix must store an intersection peak on all segments meeting at that peak. Intersections can be noted by the user on the C.R.T. but no internal mechanism notes the intersection. An examination scheme where a marker (cursor) is moved along segments by the user's control would not allow the marker to be moved onto a segment being intersected in this case.

In summary, two of the four objectives are achieved well for the track matrix. Intersections are handled; but somewhat inconveniently.

Applying peak or track grouping to this method would have little effect on its storage size shortcoming. Solving the size problem by making each segment run end to end in one long array allows one to fix a total length limit on all segments and eliminate much of the wasted storage. Indices would have to be kept indicating where each segment begins in the array and at what time it begins. This idea is a step toward the best solution, but its lack of time ordering across time slices needs to be corrected.

GRAPH STORAGE

Since tracks can both split and merge, a graph is the ideal representation. When a graph memory representation on a computer is being considered, an associative memory (non-sequential memory) is needed. The ability to create dynamic variable while running a program is achieved on the PDP 11/45 computer via "The Graph Information Retrieval System (GIRS)[1]. In GIRS, cells with pointers can be created as needed during execution of a program. One cell consists of a node-link-node tiple.

To represent a track with GIRS, define a node to be the starts, stops, and intersections of tracks. The nodes are connected by pointers oriented in time. Two specialized nodes are reserved, the head node which points to the starts of tracks and the tail node at which the last node in tracks point. Each non-specialized node is associated with all peak data that appears between itself and the next node downward in time. Data Pointers at each such node supply the reference to an index file. In turn, the index file has information about the starting time and length of each segment of the track, the starting location of the list of peaks in the data file associated with each segment.

The graph storage method appears to be a storage by segment method, however the whole tracks are actually stored as complete units. Segments also are well defined units in the graph.

Figure #5 shows a track and its graph representation. Note that the node corresponding to the split in the track has two pointers. Since cells are created as needed, lists of pointers can be generated for each node. To make the example clearer, we will now form the track of figure #5 segment by segment as the tracker would form it, thus giving insight into the procedure by which the graph is made.

[1] Zaritsky, I.S., "GIRS (Graph Information Retrieval System) users Manual," Report DTNSRDC-79/036 (April 1979)

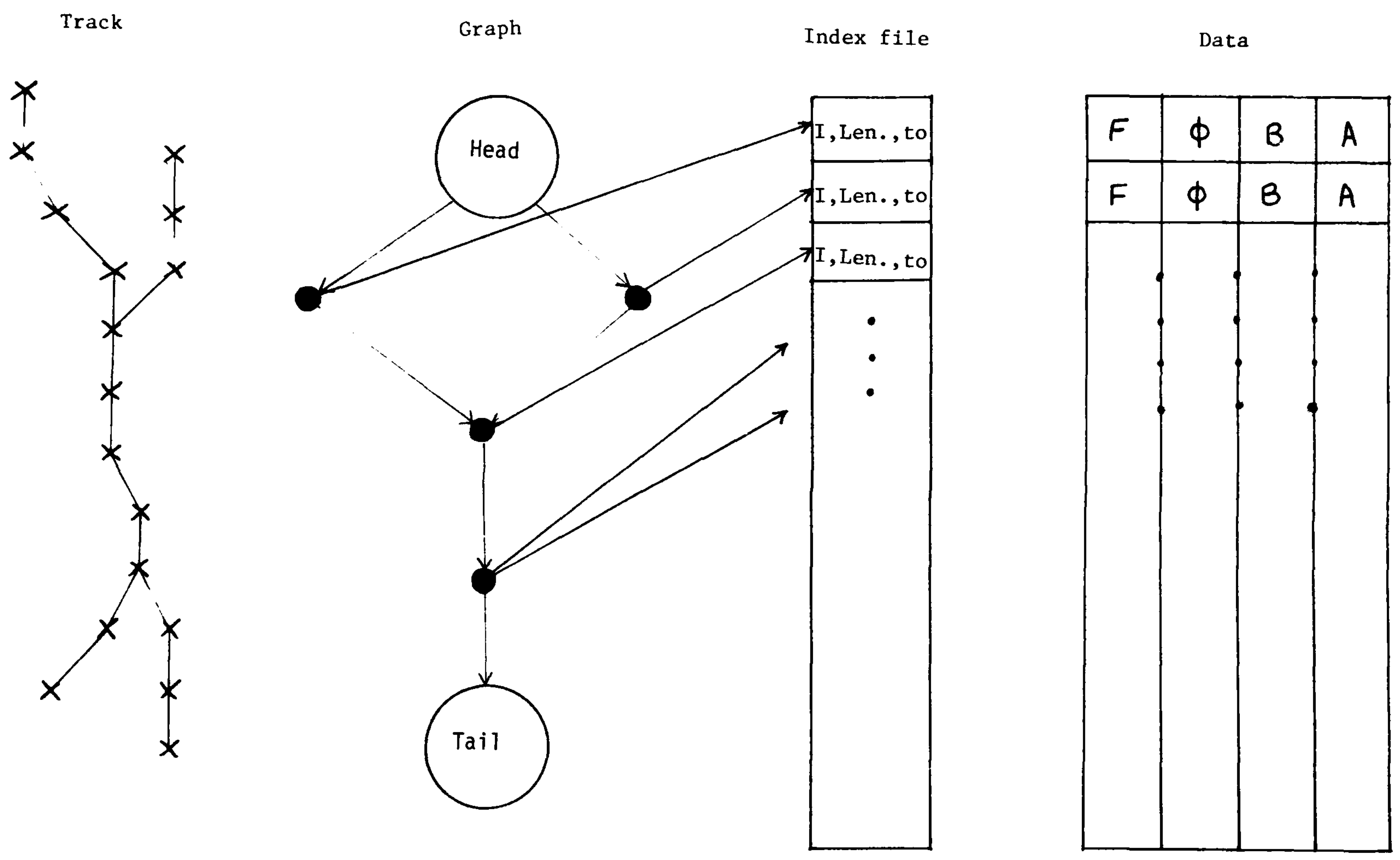


figure #5 "Graph storage"

First the tracker finds a segment and starts a new graph (see figure #5A). A node has a pointer to the index file and in turn the index there gives access to the time sequential peak data for that segment. Next, another segment is added to the track (see figure #5B). In this step, the previous entry length is divided because a junction node is inserted into the graph. A new index is entered for the second part of the divided data. Then the next segment found is also added in the data and an index entry created. Lastly, the final segment is found and the graph of figure #5 is completed.

Now the graph storage method will be analyzed with respect to the objectives.

The graph has the least storage requirements of all methods considered. The data and index files are fixed length arrays and declared in advance by a direct access unformated Fortran file procedure. The graph itself varies in size according to the amount of data to be tracked. Graph size also depends on the number of intersections in the data since extra nodes are made for intersections. By computing the storage of the fixed length arrays and estimating a typical graph with an average amount of intersections, the storage requirements can be compared for the standard example.

Index file:	$3 \frac{\text{words}}{\text{segments}}$ X 200 segments = 600 words
Data:	4,500 peaks (max. allowed) X 5 $\frac{\text{words}}{\text{peak}}$ = 22,500 words
Graph:	between 50 and 2,900 words
Total:	between 23,150 and 26,000 words

The lack of explicit ordering of tracks on time slices creates some problems with the graph. Each node contains a list of peaks in time, but these lists overlap in a random manner. To display an interval between slices n and n + 29, the whole graph will have to be traversed to search for relevant data.

Inefficiency also occurs with display examination. Traversing a given segment can be performed well since the time sequenced data for the segment can be quickly brought into core memory via the node index pointer. However, jumping to other segments covered by the same or a different node is difficult. Figure #5C gives an example of this. The user is viewing peak 4

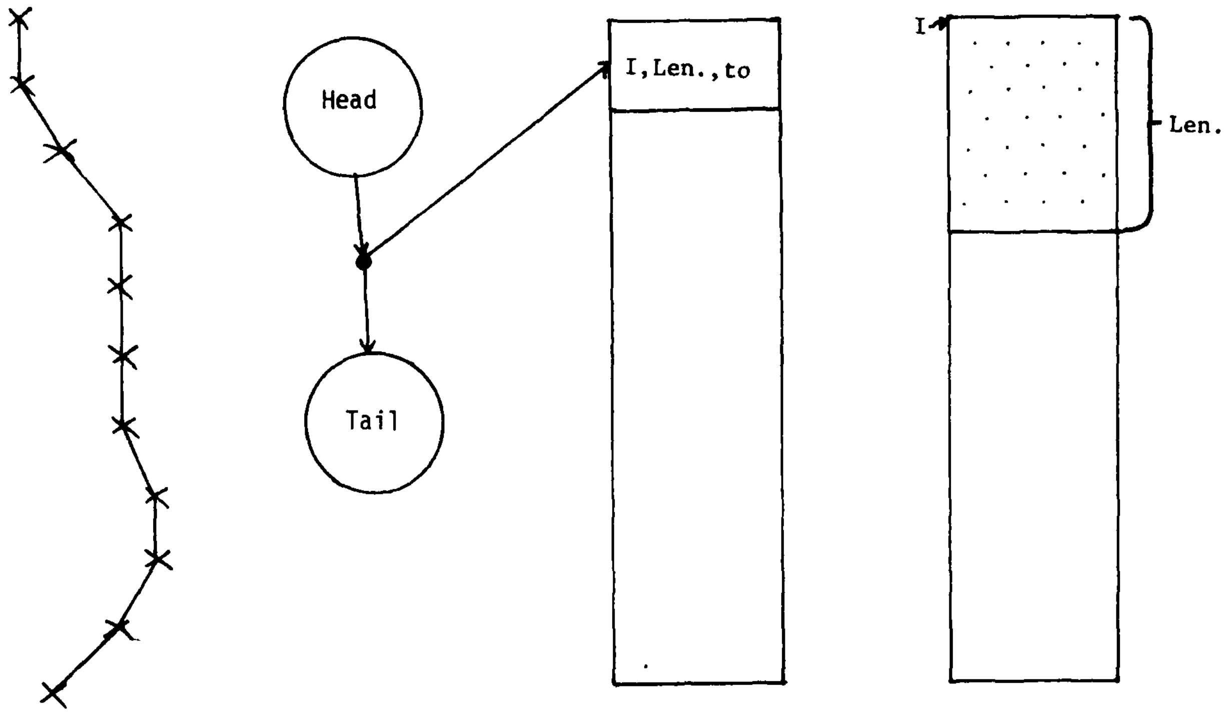


figure # 5A

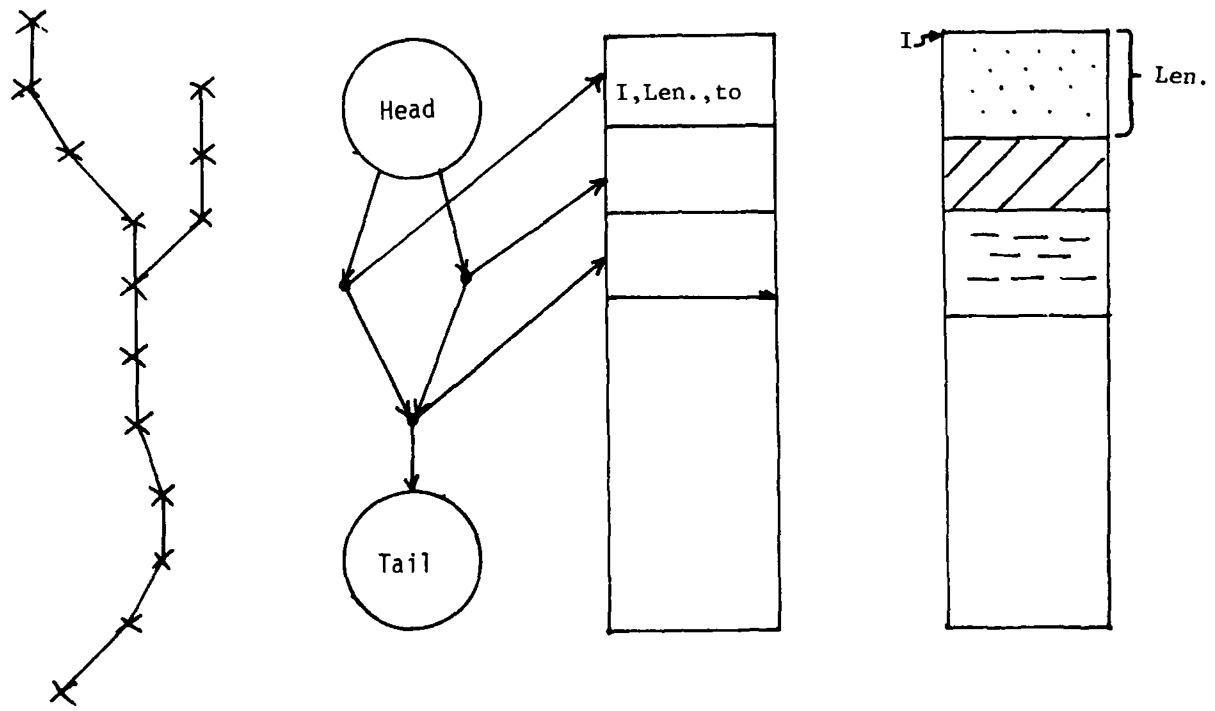
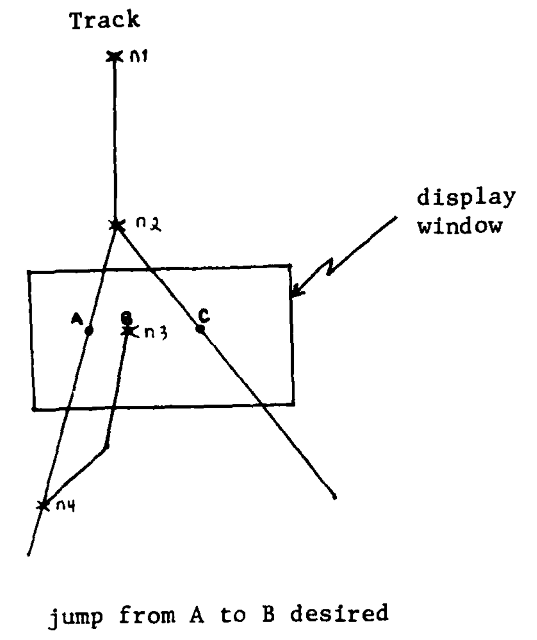


figure # 5B



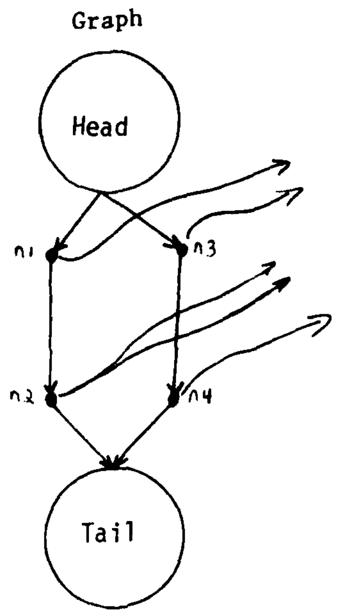


figure #5C

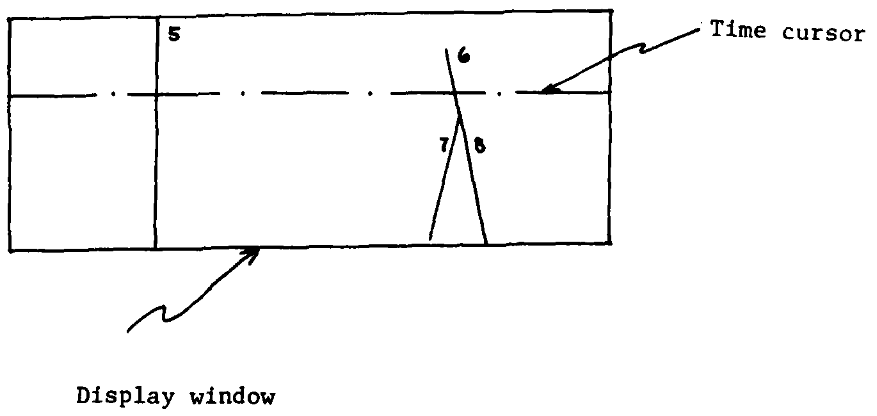


figure #5D

vector information at point A on the track display. He then wants to jump to the next higher frequency on the same time slice (Point B). If the jump mechanism moves to the segment on the indicated time slice by using the nearest node that has a segment on that time slice, point C will be reached since both points A and C are covered by the same node. Therefore, a search of the entire graph must be done in jumping from segment to segment on the display screen.

Intersections are taken care of naturally in the graph method and no problems will occur.

In summary, the graph storage method corresponds to a time space product similar to that of the Track Matrix, although here storage size is low and retrieval time is long. Before going on to the last general idea, a few variations of the graph will be mentioned.

VARIATIONS OF THE GRAPH METHOD

Peak graph

With regard to the peak-segment-track storage grouping options, one might have each peak in the track have its own node with pointers. This idea makes the graph correspond one-to-one with the track. The storage overhead associated with GIRS (four words per node link node triple) and the many nodes required for this variation causes the storage size to become very much larger than originally. The purpose of the graph storage method would be defeated with this variation.

Tagged display

One way of helping display examination of the graph structure is by placing tags adjacent to each segment on the display screen. In this scheme, the tracking routine marks each segment found in the graph with a tag number. The tag number is in a GIRS cell with a pointer to the index file. The tags are to be displayed on the screen adjacent to its corresponding segment. The user explicitly prompts the examination mechanism by supplying the tag when a new segment is to be examined. An example of this is shown in figure #5D. A potentiometer knob on the PDP 11/45 is used to move a horizontal time cursor up and down the display via the Fortran graphics library routines. The cursor is set on a time slice and the command "SEG=6" is entered on the keyboard. The segment information would be read into core and the 4 vector information for the peak marked by the cursor would appear on the screen. Other peak information on that segment would be obtained by moving the time cursor along the segment.

The tagged display idea is an attempt to help display examination only. Because of the tags, display creation presents even more problems. An algorithm that places the tags not only within bounds of the display, but at a place clearly marking the segment it belongs to is needed. The extra problems incurred from the tag idea offset the benefits. The tags also will increase storage requirements but not significantly.

Specialized links

Adding special links will help traversal of the graph. Links could be added to the graph during creation that would fix a predetermined order of traversal. The examination routine would not have to search for a new segment, but the user would have no choice in the order of segment examination. A time link set could be maintained by having time links for all slices having one or more tracks on it. The time link would point to all nodes containing segments crossing that time. Many time links would be needed causing a large storage addition offsetting the benefit obtained. Many specialized link ideas can be derived, however, storage increases and ambiguity of traversal under certain situations kept specialized links from being implemented.

TIME MATRIX

In this method, the cartesian ordered index and data files of the graph are kept but the graph itself is exchanged for the time matrix. The time matrix will have slots for each time slice possible. Only a few segments will occur over the same time range, therefore the slots will be limited in number per time slice (see figure #6). As was mentioned earlier, the track matrix wasted space because segments did not occupy but a few of the time slices allocated. Savings are made by realizing the limited number of segments that can appear on the same time slice. The peak picking algorithm that marks significant peaks in the power spectrum, shown earlier in figure #1, will pick at most 10 peaks and store them. On the average, only six to eight of the peaks on a time slice will get involved in tracks. Interpolations across time slices produced by the tracking routine add the possibility of creating extra peaks on a time slice. If the segment grouping option is used, junctions peaks will be stored on both joining segments adding an extra peak to the time matrix. Under real situations it was found that no more than 11 peaks per time slice occured in tracks. Making some approximations in order to perform statistics gives the probability of exceeding 13 peaks on a time slice very small. Therefore the time slots in figure #6 are limited to 13 peaks. The slots in figure #6 are indices corresponding to the storage group containing the peak residing on that time slice. Again, the storage group could be peak, segment, or track. The time matrix for peak oriented storage grouping would have a different number in each slot of the time matrix. The number would be the index into the data file having the 4 vector information for that peak and indices for the peaks connected above and below. A problem occurs when creating the track display. More than one peak can connect to another peak. Creating the display by following the peak indices will allow over drawing a previously displayed segment making that segment darker than others and wasting computer time. If one wanted just to display the tracked peaks as points with no connections the peak grouping method is fine. The user could not see the connected peaks, however a cursor could move from peak to the next peak in the track by using the peak links stored in the data. Having to view non-connected peaks is undesirable and a better storage grouping exists.

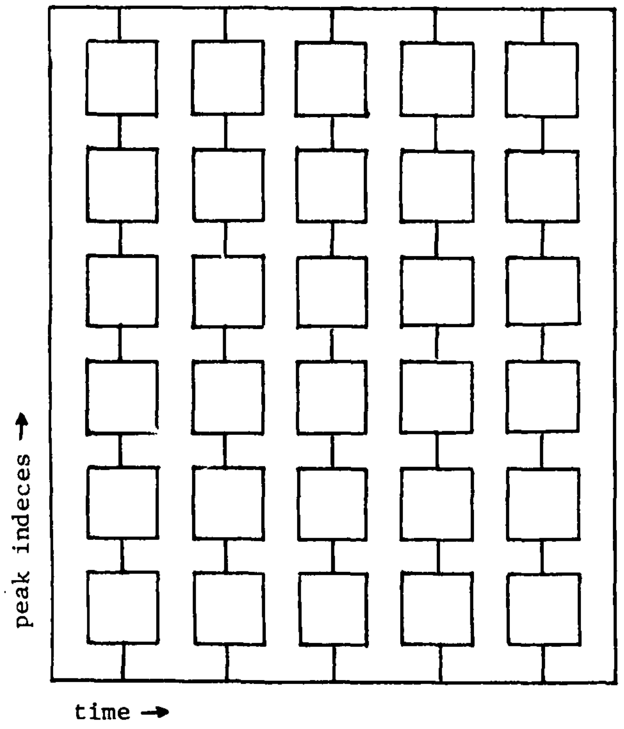


figure #6

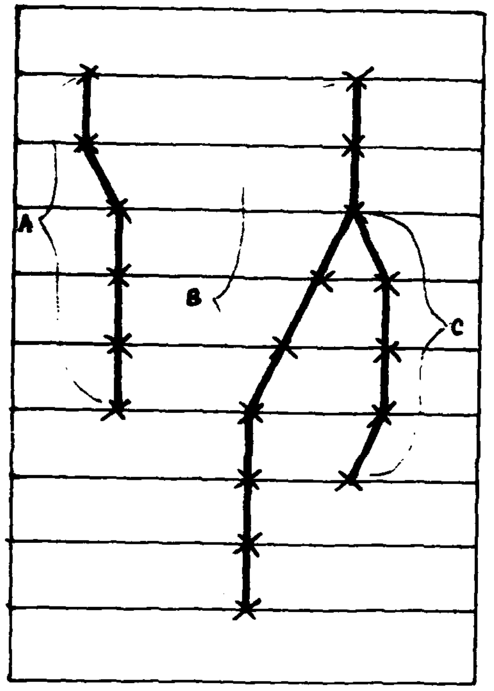
"time matrix"

If the time matrix method were to use track grouping, the slots in the matrix would refer to which track that particular peak belonged to. The data file would need no pointer indices, but a different problem arises. Because of intersections, the same number could be contained in more than one slot on the same time slice. In this case, the index in the slot will refer to the same place in data storage, but for different parts of a track! Therefore, something special would be needed to distinguish between the different parts.

Segment storage grouping is the best alternative. Each slot in the time matrix is a number referring to a particular segment. The index in the slot is used in the index file of figure #7. A "4" in a time matrix slot refers to the fourth row of the index file. Each row of the index file has five numbers pertaining to a segment. The first is a number referring to the place in the data file where peak information starts for that particular segment. Next is the time slice number for the start of the segment. Then the number of peaks for the segment is given followed by segment numbers for upward and downward intersections. A zero entry represents no intersection. The data file contains sequential 4 vector information for each peak.

The example in figure #7 will now be explained. The tracking algorithm first finds segment A. Each slice in the time matrix pertaining to segment A is updated with the number one (first segment found). The first row of the index file is filled in with the appropriate numbers and the data file has the peak information added. Segments B and C are then found and the files updated. Note segment C joins segment B at B's beginning peak. This is noted in the third entry of the index file. The data file in figure #7 contains some artificial numbers for the first few peaks as an example. The data file has a limit to the number of peaks. Though each time slice has room for 13 peaks, only six to eight peaks on the average are involved in tracks. Allowing for a miximum of 9 peaks per time slice leaves plenty of space. The data file length limit is set at 9 times the number of time slices. In the example having 500 time slices, the number of peaks allowed is 4,500. Exceeding either the time matrix or data file limits would be noted by the

Tracks



Time Matrix

1	2			
1	2			
1	2	3		
1	2	3		
1	2	3		
1	2	3		
2	3			
2				
2				

ind.	to	len.	↑link	↓link
1	1	6	0	0
7	1	9	0	0
16	3	5	2	0

⋮

index file

F	ϕ	B	A
700	87	40	96
700	87	43	91
740	103	56	88
740	103	56	90
740	101	58	91
740	100	56	94

⋮

data file

figure #7

'The time matrix method'

tracker during storage and the effect to particular parts of the data could be avoided.

Storage requirement for the time matrix method with segment grouping are computed as follows:

$$\begin{array}{rcll}
500 \text{ slices X } 13 \, \dfrac{\text{words}}{\text{slice}} & = & 6{,}500 \text{ words} & \\
+ \; 200 \text{ segments X } 5 \, \dfrac{\text{words}}{\text{segments}} & = & 1{,}000 \text{ words} & \\
+4{,}500 \text{ peaks X } 5 \, \dfrac{\text{words}}{\text{peak}} & = & \underline{22{,}500 \text{ words}} & \\
 & & 30{,}000 \text{ words} & \text{TOTAL}
\end{array}$$

Note: track matrix method -- 500,000
graph method -- 23,150 to 26,000 words

The small amount of extra storage space over the graph method has bought much time ordering in the time matrix method. The sequential data storage and easy indexing gives efficient display creation and examination. Figure #8 shows a display and the cursor used in examination. The display screen presents 3 tracks (4 segments) and below this the 4 vector values for the peak marked by the cursor (asterisk). A buffer of the same depth as the display screen (30 slices) is maintained in core memory. The peak data for the vertical movement is used to specify which peak on the segment will be examined. If a disjoint segment is to be examined, the horizontal knob is turned clockwise for a right jump and counter-clockwise for a left jump. Using the time matrix, the examination mechanism fills in the data buffer for the segment desired. If one wishes to examine a segment joined to the top or bottom of the current segment, the cursor is moved across the junction using the vertical knob causing an implied jump with the segment number being supplied by the up link instead of the time matrix.

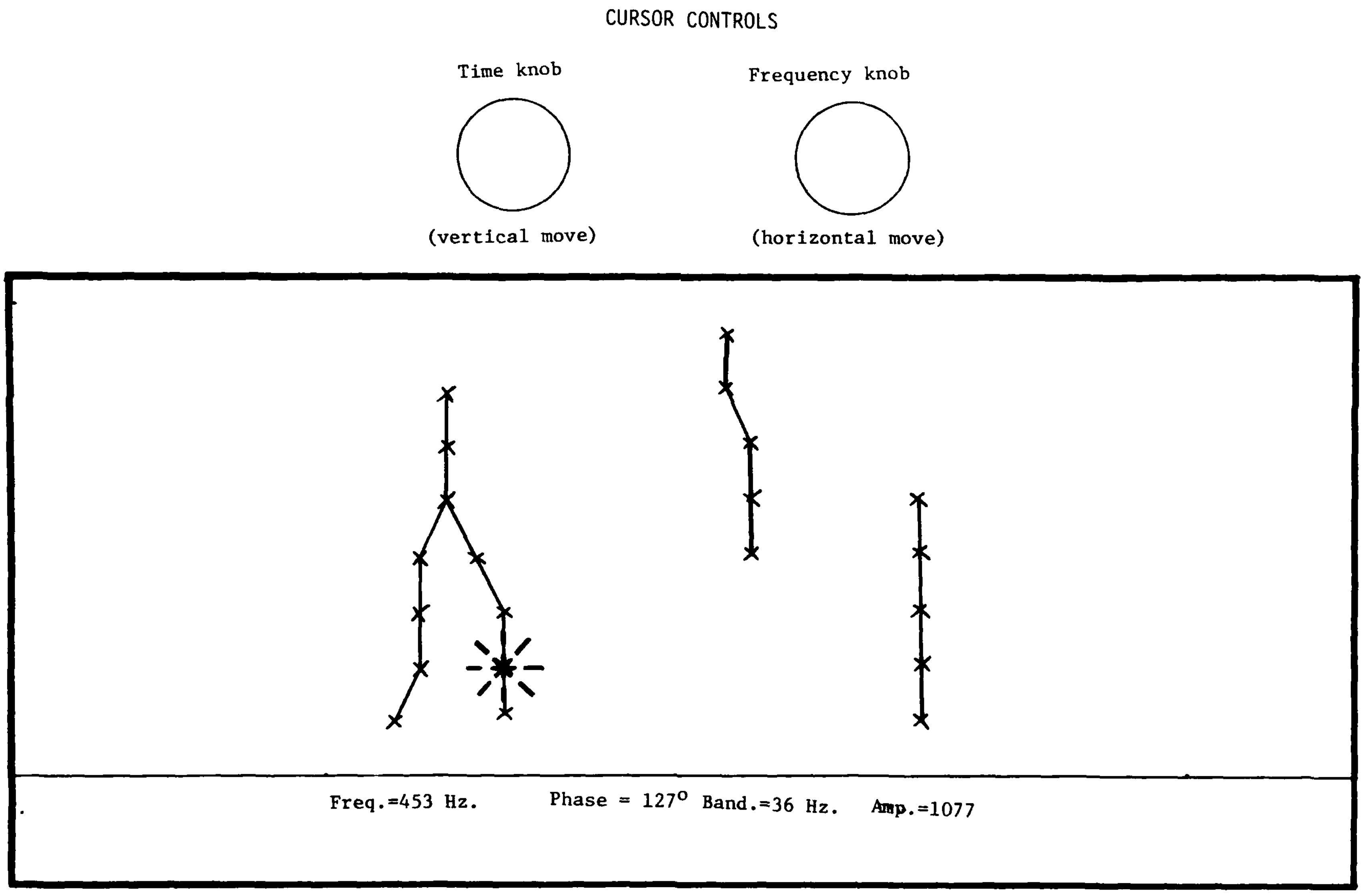


figure #8

'Track display with cursor'

SUMMARY

The time matrix method meets all the objectives for the Frequency Track storage problem and therefore is to be used as the optimum storage method.

INITIAL DISTRIBUTION

Copies	
12	DTIC

CENTER DISTRIBUTION

Copies	Code	Name
1	18/1808	G. Gleissner
1	1802.2	F. Frenkiel
1	1802.4	F. Theilheimer
1	1805	E. Cuthill
2	1809.3	D. Harris
1	182	A. Camara
20	1824	C. Slominski
1	184	J. Schot
1	185	T. Corin
1	187	M. Zubkoff
1	189	G. Gray
1	522.1	Library (C)
1	522.2	Library (A)