United States Department of the Interior
Geological Survey

AGRAM: a Series of Computer Programs for Processing Digitized Strong-Motion Accelerograms

Version 2.0

Compiled by
April Converse
01 July 84

Open-File Report
84-525

TABLE OF CONTENTS

PREFACE

This report describes the computer programs that are used by the U.S. Geological Survey for processing digitized strong-motion accelerograms. The report is primarily a user's guide for members of the USGS, but it will also inform organizations outside the USGS about the programs that have generated the data described in the strong motion data reports published by the USGS.

The programs are still in the development process; many of the non-standard options they provide have yet to be evaluated and many other features intended for the programs have yet to be implemented. Many of the possible future modifications are mentioned in the report. With continued support and further development, however, the programs will evolve into a comprehensive series of relatively transportable FORTRAN 77 computer programs available to any investigators who require strong-motion data-processing software.

This is the second version, version 2.0, of the report. Change bars like those in the margin of this page indicate sections of the report that have been added or changed significantly since version 1.0 was published. New programs, IOMPLT, FASPLT, ROTATE and BBDATA, have been added and the interface beween the BUTTER program and its user has been extensively revised.

The programs and report will continue to change as their development progresses. This version of the report will become outdated, just as the last version did, as the capabilities of the programs continue to expand.

01 July 1984

April Converse
U. S. Geological Survey
Mail Stop 977
345 Middlefield Road
Menlo Park, CA, 94025

telephone: (415) 323-8111
extension 2881

or FTS 467-2881

ACKNOWLEDGMENTS

This report and the computer programs it describes are compilations of techniques and recommendations devised through cooperative efforts of many members of the USGS. A. Gerald Brady, William Joyner, Peter Mork, Virgilio Perez and Michael Raugh have all made substantial contributions.

Many of the techniques used in these programs, even the names of some of the programs, were adapted from a series of programs that were originally developed at the Earthquake Engineering Research Laboratory of the California Institute of Technology during the years 1968 to 1972.

This work was funded in part by National Science Foundation Grant CA-114, under interagency agreement with the USGS through 1982. Support for subsequent development has been assumed by the USGS.

CHAPTER 1

Introduction

1.1 Overview

AGRAM is a comprehensive series of computer programs for processing strong-motion data. Although the programs were established to process strong-motion accelerograms that have been digitized by the automatic trace-following laser equipment at IOM/TOWILL corporation, they will accept data from other sources as well. The programs now accept data that have been recorded in digital form by a Sprengnether DR100 recorder and they will eventually accept data that have been acquired by other types of digital recorders or by manual digitization.

The contents of a tape written at the digitizing facility are processed at the National Strong Motion Data Center (NSMDC) at the USGS offices in Menlo Park, California. First, the IOMTAP program is used to read the tape and translate its contents. Next, the BUTTER program is used to rejoin separately digitized frames of an accelerogram into one continuous record. The SCALE program then scales the data to represent time and acceleration rather than digitizer units. Next, HIFRIC interpolates the data, applies an instrument correction and filters high frequencies from the data. CORAVD integrates acceleration to obtain velocity and displacement and optionally performs a linear base line correction or filters out long-period content. The FASPLT, PHASE3, and RSPECT programs perform spectral analyses of the CORAVD results.

1.2 Data Sources

1.2.1 Analog Records

Most of the strong-motion recording instruments that provide the raw data to be processed by the AGRAM programs discussed in this report are analog accelerographs that record on 70mm film. Older accelerographs record on 12-inch paper and a few multi-channel accelerographs record on 7-inch film. Strong-motion accelerographs do not record continuously but begin recording only after earthquake motion has triggered the instrument. Recorders are usually set to trigger with approximately .01*g* in the vertical direction and are designed to be up to speed in, at most, 0.1 second.

Most records contain three accelerometer traces, one or more reference (or "fixed") traces, and one or more time traces. The three accelerometer traces correspond to orthogonal components of motion. On most records, the first (topmost) accelerometer trace represents horizontal motion, the second represents vertical motion, and the third represents horizontal motion perpendicular to that of the first trace. The reference traces are produced by "fixed" mirrors rigidly attached to the accelerograph frame; they are used to correct for film distortion or transverse slippage of the film as it moved through the accelerograph. Time traces show a pulse approximately every half centimeter, the distance between pulses representing a half-second time interval. The time traces are generated by an internal timer that is more accurate than the recording speed is constant. Some accelerograms also show WWVB time code traces with which the date and time of motion can be determined. All these analog traces (except the WWVB trace) must be digitized before they can be processed by the computer programs.

Digitization is currently done at IOM/TOWILL corporation in Santa Clara, California with automatic trace-following, laser scanning equipment. The laser scanner provides much better resolution than can be attained with the manual digitization that was used in the past, but it can digitize at most a 10 square centimeter area at one time. Between 500 and 800 points per centimeter of trace length are digitized with a resolution of 1 micron (10^{-6} meter) and an RMS error of the order of 10 microns. Records longer than 10 centimeters are divided into two or more "frames" that are digitized independently. Transverse "butt" lines are scribed onto such records as dividers between frames. All the continuous traces on each frame are digitized with half an inch of overlap beyond the butt lines and the butt lines are also digitized. The overlap and butt lines are used by the BUTTER program to rejoin the separately digitized frames.

The scanning laser automatically digitizes the traces once an operator starts it on each trace in turn. Manual intervention is required for handling situations requiring human decision such as intersecting traces, gaps or faint areas in the trace, or dirt and scratches on the film. The time trace is also measured manually at the leading edge (or any repeating clearly defined corner) of each time mark. The laser scanner measures on the order of 6,000 points (12,000 numbers) per trace per 10-centimeter frame.

The tape is read at NSMDC by the IOMTAP program. It converts each number given on the tape from the internal binary format used by the computer at IOM/TOWILL to the internal binary format used by the computers at NSMDC so subsequent programs can conveniently process the data. The sequence of programs used for routine processing of data that has been digitized at IOM/TOWILL is:

```
IOMTAP
BUTTER
REFORM  (this step may be unneccessary soon)
SCALE
HIFRIC
CORAVD
FASPLT
PHASE3  (a new program, RSPECT, will replace PHASE3 soon).
```

The support programs TSPLOT, BBFILE, BWRITE, ROTATE and REFORM can be used at various stages in this process. Other programs, such as those used to perform the response analysis calculations in references [17], [18] and [19] may be added to the AGRAM series in the future.

1.2.2 Digital Records

Many recently developed instruments record directly onto digital magnetic tape. These instruments acquire data continuously, always saving the last few seconds of data in a buffer so that motion preceding instrument triggering can be captured with the rest of the record.

The computer programs described in this report were established to process digitized analog records, but they can be used to process digitally recorded data as well, once the data has been read and converted to an appropriate format. The programs have been used to process data from Sprengnether DR100 instruments and will be modified as necessary in the future to accommodate data from other digital recording instruments. DR100 recorders have been used with velocity transducers and with force-balance accelerometers. At NSMDC, DR100 tape cassettes are read using the program DR11K, next, the several components of motion that were recorded together are separated ("demultiplexed") by the CRTAPE program, and the data are then reformatted by the program named DR100. These three programs, and many others used to manipulate DR100 data, are not part of the AGRAM series and are not described in this report. Some information about them is given in the PDP machine at NSMDC in the text files at DR0:[2,2]DR11K.DOC, DR0:[2,2]CRTAPE.TXT and DR0:[2,2]DR100.DOC. Edward Cranswick, Charles Mueller, and Jon Fletcher (all of the USGS) can provide more information about the processing used for DR100 data.

Digital recorded data formatted as in DR100 output files may enter the AGRAM processing at HIFRIC or CORAVD. Since the data in DR100 files is evenly sampled, the HIFRIC step may be bypassed if frequencies outside the range in which the instrument response is flat are beyond the range of interest. HIFRIC will process digital data that requires instrument correction, but the correction is for a simple damped oscillator operated as an accelerometer. Users must consider whether or not such instrument correction is appropriate for the instrument that recorded their data. If not, a new program or new HIFRIC options will be required to perform the instrument correction.

The sequence of programs for processing digital recorded data is:

DR11K
CRTAPE
DR100
HIFRIC
CORAVD
FASPLT
PHASE3 (or RSPECT).

1.3 Background

The AGRAM programs are revisions or rewrites of the BUTTER and PHASE1 through PHASE6 programs that were installed at the Lawrence Berkeley Laboratory (LBL) computing center long before the National Strong Motion Data Center (NSMDC) existed. The LBL versions of the programs were used from 1974 through 1981 in preparing the strong motion data reports published by the Seismic Engineering Branch of the USGS. The PHASE1 through PHASE4 programs were originally developed at the Earthquake Engineering Research Laboratory of California Institute of Technology although modifications were made by Seismic Engineering Branch to upgrade procedures and to adapt the programs for use with the computers at LBL. PHASE5 and 6 were developed by Virgilio Perez (USGS). BUTTER was developed by Gerald Brady (USGS) and by W.R. Roseman of IOM/TOWILL.

All the programs were revised in order to transfer them from the CDC 7600 computer at LBL to the PDP 11/70 at the NSMDC, and although they were modified considerably in the process, they retained the same names. Recently, most of the programs have been revised to use a new data file format, to separate the processing functions into smaller and more independent programs, to use new processing algorithms that have been developed by Michael Raugh (formerly at the USGS) and others, and to move the programs from the PDP 11/70 computer to the VAX computers at NSMDC.

Those programs that have been revised to the point where they produce blocked binary data files rather than card-image text files have been renamed as a reminder that the programs are now significantly different from their PHASE counterparts at LBL or CalTech. SCALE has replaced PHASE1, HIFRIC and CORAVD have replaced PHASE2, FASPLT has replaced PHASE4, and RSPECT, when it has been completed, will replace PHASE3. The PHASE5, PHASE6 and other response analysis programs developed by Virgilio Perez may also be reorganized and added to the AGRAM series in the future. A brief discussion of the differences between the AGRAM programs and their PHASE counterparts is given in appendix C.

1.4 Running the programs

The AGRAM programs may be used with either the VAX 11/750 or the VAX 11/780 computers at NSMDC. All the instructions given in this report are presented as though the programs are running on a VAX machine. The $ prompt given by the VAX operating system to an interactive user and the structure of the file names shown in this report would have different counterparts on another machine. VAX file names take the form

```
device:[directory.subdirectory]name.suf
```

where device identifies an input/output device such as a disk. Device names of MTA0: (a tape drive) and PUB1: (a disk) are used in this report. Directory is a root-level directory in a disk storage hierarchy, and subdirectory is a subdirectory in the storage hierarchy. The directory at PUB1:[AGRAM.TESTDATA] is often used in the examples in this report. Name is the prefix portion of a file name (9 characters long or less) and suf is the suffix portion of a file name (3 characters). Many of the file names shown in this report specify just the name and suf fields, relying on the VAX conventions for supplying defaults for the other fields in the file name. Here are two sample file names: PUB1:[AGRAM.TESTDATA]EDA.R01 and TEMPORARY.A01.

In order to use any of the AGRAM programs, one must first execute the commands in the file at PUB1:[AGRAM]AGRAM.DEF. To do so, use:

```
$ @PUB1:[AGRAM]AGRAM.DEF
```

If you will be using the programs frequently, consider including that statement in your LOGIN.COM procedure. Among other things, the commands in AGRAM.DEF redefine the HELP command to provide on-line help about the AGRAM programs in addition to the various other topics the HELP command normally provides. To get AGRAM information from the on-line help facility, type:

```
     $ HELP AGRAM
     $ HELP AGRAM TOPICS
     $ HELP AGRAM START
     $ HELP AGRAM CHANGES
     $ HELP AGRAM BUGS
     $ HELP AGRAM CODE
 or  $ HELP  <any of the AGRAM topics shown in the
             HELP AGRAM TOPICS display>.
```

Most of the AGRAM information available from the HELP command is also contained in this report, but the "$ HELP AGRAM CHANGES" section will provide information on changes and additions that will have been made to the AGRAM programs after this report has been printed.

To run any of the AGRAM programs, type the name of the program followed on the same line by the run parameters the program requires. Several of the programs, BUTTER, BWRITE and PHASE3, do not yet require that the run parameters be given on the same line; these three programs get their run parameters by prompting the user although they will eventually be changed to receive their parameters, like all the other AGRAM programs, from the same command line with which they were invoked. If no run parameters are given on a command line that invokes an AGRAM program (other than BUTTER, BWRITE and PHASE3) the program will display instructions informing you about the run parameters the program requires. In the instructions displayed by the programs and in the instructions given in this report, items that the user must provide are shown in brackets. Items in square brackets ([]) are required, items in angle brackets (<>) are optional. Items in quotes ("...") are literals. Note that you do not type the brackets or the quotes when providing such items.

The command lines for these programs are arranged so that output file names are listed on the left-hand side of an equal sign and input file names are listed on the right-hand side of the equal sign. The file names may be abbreviated: if the directory is not specified for the first file name given on the command line, the user's current working directory is assumed; if the directory or directory and file name prefix is not specified for any but the first name given, the directory and prefix used for the preceeding file name is assumed. If neither the first file name prefix nor the input file name prefix is specified, "TEMPORARY." will be used as the prefix for all the files read and written by the program. If a file name suffix is not specified, a default will be selected that depends on which program and which file is involved.

Any other run parameters that a program requires are listed on the command line after the file names.

Commands that do not fit on one line can be continued by using a hyphen (-) as the last character on the line to be continued.

1.5 Warnings

The file names given in command lines must not include version numbers. If this restriction causes problems, it can be removed. The file names must not include wildcards ("*") either.

File names must contain either at least one alphabetic character or a directory specification. If not, the command line interpreters will assume the name is a numeric run parameter, not a file name.

Most of the AGRAM programs will select default output data file names if the user does not specify them on a command line. If output data file names are not explicitly specified on a command line, however, another option provided by many of the AGRAM programs, that which allows the user to specify an output text file in addition to the output blocked binary data files, may <u>not</u> be used. The output text files recieve run messages and diagnostics that would otherwise be displayed at an interactive user's terminal or written to a batch job's log file. When such a text file name is specified on a command line, it must be given after the explicit specification(s) of the output data file name(s).

CHAPTER 2

Data Files

2.1 Overview

The data files processed by the AGRAM programs conform to a standard file organization intended for all time-series disk files at the NSMDC. These files are referred to as "Vinton-Fletcher blocked binary data files" or some abbreviation thereof. Their contents are in binary form and are arranged as a series of fixed-length, 512-byte blocks that may be accessed randomly or sequentially. Each file contains time-series data for a single component of motion and a single type (acceleration, velocity or displacement) of motion. The first few blocks of each file are reserved for auxiliary information that supplements or describes the time-series data in the remainder of the file. Refer to appendix A for description of the content of the header blocks.

Input/output operations are much faster with the blocked binary files than with the card-image text files that had been processed by programs used in the past since there is no translation back and forth between internal (binary) form and external (character code) form when reading or writing binary files. Blocked binary files also require less disk space than do card-image text files. Blocked binary files may be processed by input/output software that is more efficient than that provided through standard FORTRAN i/o statements. The i/o can still be done through standard FORTRAN, however, and a standard FORTRAN version of the i/o software is included in the AGRAM distribution tape.

Since the data is in binary form, the blocked binary files must be formatted somehow before their contents can be displayed, processed with a text-editor, or transfered to another computer. The BBFILE and REFORM programs do this formatting although they aren't as versatile as they should be. BBFILE prepares a text-file dump of all or part of a blocked binary file. REFORM reformats between various types of text data files and blocked binary data files.

2.2 Raw Data

Raw data comes to the NSMDC from the digitizing facilities or recording instruments in a variety of formats and on a variety of media. IOM-TOWILL digitized data is written on magnetic tape, DR100 data is recorded on tape cassettes, manually digitized data from the NSMDC digitizer is written on a floppy disk, and we still occasionally receive data on punched cards. Each type of incoming data requires a special program or programs to read and translate the data. Gerald Brady (USGS) or Willian Roseman and Robert Pettit at IOM/TOWILL can provide information about the format of the IOM-TOWILL tapes; these tapes are read and translated by the MTDUP program. Jon Fletcher and Lawrence Baker (both at USGS) can provide information about the format of the DR100 cassettes; they are read by the DR11K program and are translated by the program named DR100. John Boatwright (USGS) can provide information about the format of the floppy disks prepared by the digitizer at NSMDC.

2.3 DR100, HIFRIC and CORAVD files

Time-series data in files generated by the DR100 program are integers (256 2-byte values per block) and time series data generated by the AGRAM programs are reals (128 4-byte values per block). DR100, HIFRIC and CORAVD output files contain acceleration, velocity or displacement values only: time is at even intervals, the time interval being stored in a header block.

For some DR100 files, some of the recording instrument characteristics that were once intended to be stored in header blocks are now stored in a separate, "DBS", file. Edward Cranswick and Charles Mueller (USGS) oversee the DR100 and DBS files at NSMDC. Instrument characteristics are still stored in the header blocks of AGRAM files.

2.4 BUTTER and SCALE files

The time-series data in the output files generated by BUTTER and SCALE include the abscissa for each sample since the data are not at even intervals. Data are real and occur as a series of (x,y) pairs. In BUTTER files, x and y are in digitization units (=microns if the digitization was done at IOM/TOWILL). In SCALE files, x is in seconds, and y is scaled, uncorrected instrument response in cm/sec/sec offset by a constant value (the constant being in a header block.)

A BUTTER output file contains data for all the traces on a digitized record, but a SCALE output file contains the data for just a single accelerometer trace.

2.5 File Names

Names for most of the time series files on the disks at NSMDC are selected to indicate which data the file contains. The DR100 file naming convention is quite specific since there are thousands of DR100 files that the file names must uniquely identify. File names used with AGRAM data are less specific, since there are only relatively few AGRAM files on the disks at any one time. (AGRAM data files are so large that they are removed from the disks once processing is complete and are archived on tape.) Twelve characters are allowed in disk file names used by the computers at NSMDC. The names are in two parts, a nine-character prefix and a 3-character suffix, separated by a period.

There is no strict convention for selecting file name prefixes for AGRAM files although the same prefix is used for all files representing the same record and that prefix is usually an abbreviation that indicates the source of the data. In the examples shown in appendix B, for instance, the prefix PVB was used as a reminder that the files contain data from the Pleasant Valley pumping plant basement. The suffixes used for AGRAM data files follow a convention that allows us to distinguish from which stage in the processing a file comes. The first character of a suffix indicates what sort of data is in the file, and the second and third characters are numeric digits that indicate which trace from among the several traces that were recorded together is contained in the file. SMA records usually consist of traces from three orthogonal transducers with the first and third traces for horizontally oriented transducers and the second trace for a vertically oriented transducer. But things are not always so simple! There are remote recorders in the field that can record up to 16 traces on a single record. Transducers are not always arranged in orthogonal triplets, either.

Here follows a table relating the AGRAM file name suffixes and the programs that produce and read those files. The TINKER program shown in the table is not discussed elsewhere in this report; it is primarily used to process DR100 data.

suffix	producing program	file content	reading program
.IOM	IOMTAP	Raw digitized data	BUTTER
.BOU	BUTTER	Butted IOM digitized frames. (text)	REFORM
.BUT	REFORM	" . (blocked b)	SCALE
.R01	SCALE	Uninterpolated instrument response.	HIFRIC
.G01	HIFRIC	Interpolated, instrument corrected case acceleration.	CORAVD, or TINKER
.I01	HIFRIC	Interpolated, uncorrected instrument response.	TINKER
.A01	CORAVD	Baseline corrected acceleration.	FASPLT,RSPECT, PHASE3 or TINKER
.V01	"	Velocity.	"
.D01	"	Displacement.	"

DR100 file names do not need to distinguish among several levels of processing, since only one level (that corresponding to the AGRAM .A files) is saved on the disk. DR100 File names have the form JJJHHMMSC.STA, where JJJ is the Julian day, HH is the hour, MM is the minute, S is a letter A through T, C is the component number, and STA is the 3 letter station name. The second, S, is represented as the letters A-T, each letter representing a 3 second interval. The component number, C, is 1,2, or 3 for acceleration; 4, 5, or 6 for velocity; and 7, 8, or 9 for displacement.

CHAPTER 3

IOMTAP
Tape Reading Program

3.1 Overview

IOMTAP reads several consecutive files from a magnetic tape prepared by the digitizing facility at IOM/TOWILL corporation, converts the data from the internal binary format used by the computer at IOM/TOWILL to the internal binary format used by the VAX computers, then writes the data to a disk file in a form that can be read by the BUTTER program. An IOM/TOWILL tape usually contains data for several strong-motion records, so the IOMTAP program must separate the data belonging to separate records into separate disk files. Each record may have been digitized in several frames with each frame represented on the tape as a separate file. When reading a tape, the user must indicate to the IOMTAP program which files from the tape shall be transfered to the disk file. The user must refer to the list that IOM/TOWILL provides along with the tape to decide which tape files belong with which record.

In the past, the tapes were read by the MTDUP utility program on the RSX computer, but this function is now performed by the more specific and more portable IOMTAP program. In addition to reading tapes, IOMTAP can also be used to read and translate files that were originally prepared by MTDUP. The BUTTER program can no longer read the old MTDUP files directly as an earlier version of BUTTER could; any MTDUP files that still need to be processed must be translated by IOMTAP before BUTTER can process them.

3.2 Running IOMTAP

Take the following steps to use IOMTAP with a tape.

a) First, allocate a tape drive to the terminal session. Use the show and allocate commands as follows.

```
$ SHOW DEV MT:     (To learn which tape drives are free)
$ ALL [drive]     (Allocate a drive to your job)
```

The tape drives at NSMDC are marked as MTA0: or MTA1: on the front of their cabinets and in the list you receive from the

SHOW DEV command. Substitude MTA0: or MTA1: wherever [drive] is shown in these examples.

b) Hang the tape:
 -Remove the yellow write ring, if there is one, in the center of the tape reel.
 -Open the latch in the center of the spindle on the tape drive.
 -Push the tape reel around the spindle.
 -Close the latch and turn the reel a quarter turn or so to make sure the tape reel is all the way on the spindle.
 -Press buttons on the right-hand side of the drive:
 POWER
 LOAD/REWIND
 ONLINE

c) At the terminal, use the mount command to inform the VAX operating system that the tape is "foreign" with large, 4000 byte, records.

```
$ MOU [drive]/FOR/BLOCKSIZE=4000
```

d) Run IOMTAP:

```
$ IOMTAP [outfile],<msgfile>=[drive],[#1]-[#2]
```

Where [outfile] is the name of the output disk file, <msgfile> is the name of an optional disk file to receive the run messages and diagnostics that would normally go to the users terminal, [#1] is the number of the first file to transfer from the tape to the disk file, and [#2] is the number of the last file to transfer. Example —

```
$ IOMTAP NEWDATA.IOM=MTA1:,1-6
```

e) Dismount the tape and deallocate the tape drive:

```
$ DISM [drive]     !(or DISM [drive]/NOUNL)
$ DEAL [drive]
```

f) Remove the tape from the tape drive.

3.3 Running IOMTAP with a disk file

To use IOMTAP with an MTDUP disk file, type:

```
$ IOMTAP [outfile],<msgfile>=[in file]
```

3.4 IOMTAP Limitations and Possible Futute Modifications

It would be helpful, particularly when dealing with a tape in which digitization errors are suspected, to have a plotting program that would plot the contents of a file produced by IOMTAP. There is no such program yet, but there should be soon. The new program will be named IOMPLT.

As IOMTAP is reading a tape, it tells the user how many 8-bit bytes of data it has found in each block read from the tape. The list that IOM/TOWILL provides along with the tape, however, indicates the number of 16-bit "words" that are in each tape block. It would be more convenient if the list IOMTAP generates corresponded with the IOM/TOWILL list and counted 16-bit items rather than 8-bit items.

CHAPTER 4

BUTTER
Frame Butting Program

4.1 Overview

BUTTER butts together the separately digitized frames of a strong-motion record that has been digitized at IOM/TOWILL. BUTTER reads a disk file prepared by the IOMTAP program and prepares an output file suitable for input to the REFORM program. Once the data has been REFORMatted it can be used as input to the SCALE program.

A new version of BUTTER should be available soon. The new BUTTER, unlike the present version, will have a command line interpreter like all the other AGRAM programs, and it will prepare its output files in blocked binary form so they will no longer need to be sent through REFORM before they are processed by the SCALE program. Refer to the information given by the on-line help facility to learn the current status of the BUTTER program. To do so, type "$ HELP BUTTER" or "$ HELP AGRAM CHANGES".

4.2 Running BUTTER

To run the present version of BUTTER, type:

```
$ BUTTER
```

BUTTER will prompt for all the parameters it needs. If all the prompts are answered with a carriage-return, a sample BUTTER run will result. The sample processes the first frame of the basement record from the Pleasant Valley pumping plant during the 02may83 Coalinga main shock. In response to the first prompt from BUTTER, the name of a text file containing an already established set of answers to the prompts (all but the first prompt, that is!) may be given. Such a text file is available as an example at

```
PUB1:[AGRAM.BUTTER]TRY6.TXT.
```

This example processes all six frames of the Pleasant Valley pumping plant record. A sample command deck for running BUTTER in indirect command mode is at

PUB1:[AGRAM.BUTTER]TRY6.COM,

and a sample batch command deck is at

PUB1:[AGRAM.BUTTER]TRY6.BAT.

Comments may be given in any line of text given in response to a BUTTER prompt; an exclamation symbol (!) indicates the beginning of a comment. Lines containing nothing but comment must have the "!" in the first character position. The three sample decks (TRY6.TXT, TRY6.COM and TRY6.BAT) all contain comments that explain what the responses are for. Here follows a print-out of the TRY6.COM deck:

```
$!
$!      This is [agram.butter]try6.com
$!      It tests the BUTTER program in indirect command mode
$!      by processing all six frames of the PVbase testdata.
$!
$!      use: $ @try6.com
$!
$set verify
$on warning then goto end
$ BUTTER
 NONE                    !read run parameters from this deck
 NONE                    !want run messages to terminal, not disk
 PUB1:[AGRAM.TESTDATA]SAMPLE.IOM    != the input data file
 NONE                    !  no need for calibration file
 TEMP.BOU                != name of the output data file
 YES                     !yes, want bbf output file
! a nothing-but-comment line here, just for illustration
  -400                   != min forward step size
 YES                     !yes, butting lines removed
 NO                      !no hanning smoothing
 NO                      !no decimation
 NO                      !no sampling
 YES                     !yes, options are correct
 SDDDRRTB                !traces in the first frame
 DDDRRTBB                !              second
 DDDRRTBB
 DDDRRTBB
 DDDRRTBB
 DDDRRTB                 !traces in the last frame
 DONE                    !end of trace descriptions
 YES                     !yes, traces are correct
$ REFORM TEMP.BUT=TEMP.BOU,BUTTER
$end:
```

Note that TRY6.COM runs REFORM as well as BUTTER since the REFORMatting function will be incorporated into BUTTER in the near future and can be considered as though it were already a part of BUTTER.

When a command line interpreter has been added to BUTTER, the program will be run by typing one of the following.

a) Just to get instructions —

```
$ BUTTER
```

b) To get BUTTER to prompt the user for run parameters just as the present version of BUTTER does now —

```
$ BUTTER PROMPT
```

c) To run BUTTER with standard run parameters —

```
$ BUTTER <outfile>,<msgfile>= [IOMTAP file],[trace type list]
```

<Outfile> is a name for the output file. If this is not given on the command line, the default output file name will have the same name as the input file except for the suffix, which will be .BUT.

<Msgfile> is the name for an optional disk file that will receive the run messages and diagnostics that would otherwise go to an interactive user's terminal or to a batch job's log file.

[IOMTAP file] is the name of the input file, a file prepared by the IOMTAP program.

[trace type list] is a list, frame by frame, of the types of traces in the frames. Each trace is identified with the letter D, R, T, B, or S for a data trace, a reference trace, a time-tick trace, a butting line, or a trace to be skipped or ignored by the program. The trace types for any one frame are given without intervening blanks or commas and each group of letters representing one frame is separated from those representing another frame by a comma. For example, the trace type specifications shown in TRY6.COM above would be given on a command line as SDDDRRTB,DDDRRTBB, DDDRRTBB, DDDRRTBB, DDDRRTBB,DDDRRTB. The trace type identifiers must be given in the same order as the digitized data for those traces is given on the input file. Refer to the list IOM/TOWILL provides along with the data tape to learn the order of the traces on the file. The sequence of traces is usually DDDRRTB for the first and last frames and DDDRRTBB for intermediate frames.

4.3 BUTTER Limitations and Possible Future Modifications

As indicated above, the output files BUTTER produces are in text form rather than in the blocked binary form that is processed by the other programs in the AGRAM series. These text files must be sent through the REFORM program to reformat them to the blocked binary files SCALE reads. This REFORMatting step is egregiously time consuming and unnecessary (internally, BUTTER processes the data in blocked binary files, but it formats the data just for output!) BUTTER should produce blocked binary files directly except in those special situations for which text output files are required in order to modify them with a text editor.

The IOM-TOWILL equipment can digitize an area 10 by 10 centimeters. For those records that are longer than 10cm., BUTTER reconnects several separately digitized frames. The program does not yet have the capability to reassemble records that are wider than 10 cm., but it will <u>need</u> to do so to process records from the newer remote recording instruments (up to 16 accelerometer traces on a 7-inch wide record).

This version of BUTTER, unlike the one at LBL, discards all but one reference trace.

This version cannot generate a dummy reference trace properly when processing multi-frame records.

It would be useful to have a plotting program that plots the contents of a BUTTER output file frame by frame, showing butting lines, reference lines, time-ticks and data traces. There is no such program yet, but the capability may be added either to the TSPLOT program or to a completely new program named BUTPLT.

CHAPTER 5

SCALE
Scaling Program

5.1 Overview

SCALE scales digitized accelerometer traces from digitizer units to seconds and cm/sec/sec. It transfers data from reformatted BUTTER output files to files suitable for input to the HIFRIC program, separating the data into a separate file for each accelerometer trace given in the BUTTER file.

In addition to the accelerometer traces, most records contain timing marks and several fixed reference traces that are also digitized. SCALE makes longitudinal time corrections based on the timing marks (assuming that the clock runs more accurately than the transport mechanism) and corrects for possible transverse slippage by subtracting the reference trace from all accelerometer traces. Optical distortions in records that have been digitized from photographic reproductions are also minimized by subtracting the reference trace.

The results from SCALE are often referred to as "uncorrected" data in the sense that no modifications have been made that involve any hypotheses as to the character of the ground motions or instruments involved. The data have been corrected merely for uneven film transport and for transverse slippage of the film as it moved through the recorder.

The results from SCALE may be plotted with the TSPLOT program.

5.2 SCALE process for each accelerometer trace:

1) Identify the reference trace nearest the accelerometer trace being processed by comparing the differences between the ordinates of the first point in the accelerometer trace and the first point in all the reference traces given.
2) If the first point in the accelerometer trace occurs before the first point in the reference trace, extrapolate the reference trace along the line fitted through the first centimeter given in the reference trace. Extrapolate the reference trace at the end of the record too, if necessary.

3) Smooth the values of the time tick abscissas with a 1/4, 1/2, 1/4 running mean (also called a Hanning filter.)
4) If the first point in the accelerometer trace occurs before the first time tick, use the interval between the first and second time ticks to extrapolate time ticks back beyond the beginning of the accelerometer trace. Use the interval between the last two actual time ticks to extrapolate the time ticks beyond the end of the record too, if necessary.
5) Perform the following steps for each point in the accelerometer trace.
   - Subtract the reference trace as follows: Identify the two reference trace points that bracket the data point. Use linear interpolation between the two reference trace points to calculate a reference trace ordinate value at the same abscissa as the data point. Subtract calculated reference trace ordinate from the data ordinate.
   - Subtract a constant also, to avoid numeric overflow problems while calculating the mean value of the trace. Use the difference between the first ordinate of the acceleration trace and the first ordinate of the reference trace as the constant.
   - Convert ordinate value from digitzer units to cm/sec/sec. Use the sensitivity of the recording transducer (cm/g) and the digitization units per centimeter in this conversion. (The default value for digitization units per centimeter is 10,000 microns/cm which is appropriate for IOM/TOWILL digitizations)
   - Convert abscissa from digitizer units to seconds using the two time ticks that bracket the data point. Calculate the time at the two ticks using the first point in the data trace as time zero, and using the time between consecutive ticks (usually 0.5 second) given as an input parameter. Interpolate linearly to calculate the time at the data point.
6) Calculate the mean value of the entire accelerometer trace. The mean is saved in a header block in the output file and is subtracted from the ordinate of every point in the trace by the next program (HIFRIC, TSPLOT, or REFORM) to process the data.

5.3 Running SCALE

To run scale, type:

```
   $ SCALE <output file(s)>,<msgfile>=[BUTTER file], -
         [time between ticks], [sens1],[sens2],[sens3]
```

Where :
<output file(s)> are optional names for the output files. If these aren't given on the command line, the default output file names will have the same prefix as the input file and will have suffixes with 'R' as the first character and the number of the data trace as the second two characters. There are usually three output files: .R01, .R02, and .R03.

<msgfile> is the name for an optional disk file that will receive the run messages and diagnostics that would normally have gone to an interactive user's terminal or to a batch job's log file.

[BUTTER file] is the input file, a data file produced by BUTTER then reformatted by REFORM into a blocked binary file.

[time between ticks] is the time, in seconds, between the tick marks on the record. Usually 0.5.

[sens...] are the recording transducer's sensitivities in cm/g, and are given in the same order as the transducer's traces occur on the digitized record. First transducer corresponds to the top-most accelerometer trace, last transducer to the bottom-most trace.

Example:

```
   $ SCALE TEMP.=[AGRAM.TESTDATA]EDA.BUT,0.5,1.92,1.85,1.77
```

5.4 SCALE Limitations and Possible Future Modifications

If there is no reference line, SCALE uses the time ticks as a reference line. If the time ticks are missing too, SCALE could (but does not yet) use the linear least squares fit to the data as a reference line and assume the recorder progressed at a constant one centimeter per second.

SCALE would probably do better to extrapolate time ticks, when necessary, with the average of the nearest few intervals rather than just the one nearest interval as it does now.

It would be more convenient for SCALE to read the time between ticks, sensitivity values, and digitization units/cm from the header blocks in the input file rather than from the command line. These items must be put into the headers before SCALE could do so, though.

SCALE is set up to process data that has been produced by the BUTTER program. For data that has come from other sources (like manually digitized data) we must either add a few more options to SCALE or set up a BUTTER counterpart for the other data. The options would include smoothing the reference trace and subtracting out the linear least squares fit of one of the reference lines from all the traces.

CHAPTER 6

HIFRIC
High Frequency Filtering and
Instrument Correcting Program

6.1 Overview

HIFRIC applies interpolation, instrument correction and a high-cut filter to SCALE data. Options provide alternative instrument-correcting and filtering algorithms. With the standard option, HIFRIC applies a time-domain algorithm; with another option it applies a frequency domain algorithm; and with a third option the instrument correction and filter are suppressed altogether, providing interpolated, uncorrected data. The time-domain instrument-correcting algorithm was written by Michael Raugh (formerly USGS), and the frequency-domain instrument-correction algorithm was written by William Joyner (USGS).

Both correcting algorithms are based on the second-order differential equation representing motion of a viscously-damped, one-degree-of-freedom oscillator (see page 46 of reference [1]). The two algorithms produce almost identical results when applied to densely digitized data from analog film recorders and when using standard filter parameters, but the standard time-domain method runs faster than the frequency domain method.

More accurate approximations of the derivatives required by the damped harmonic oscillator equation are used in HIFRIC than the centered-difference method for differentiation that had been used in earlier programs such as PHASE2. The standard HIFIRC method incorporates Fourier differentiation (i.e., $\frac{d}{dt} e^{i\omega t} = i\omega e^{i\omega t}$) by convolution operators, which are applied in the time-domain, to approximate the first and second derivatives required. The alternative HIFRIC method applies Fourier differentiation in the frequency domain.

With the time domain algorithm, the data are interpolated to an even sampling of 600 samples per second (sps) which is the approximate sampling density provided by the IOM/TOWILL digitizing equipment. Next, the frequencies from zero to Nyquist (300 Hz), are divided into six equal-width bands. The first band, from 0 to 50 Hz, is that in which instrument correction is performed. The second, from 50 to 100 Hz, is a transition band providing a cosine taper from full to zero frequency response. The remaining bands, from 100 to 300 Hz, are assigned zero frequency response. After filtering, the data are decimated by removing 2 out of every 3 points, reducing the sampling density from 600 sps to 200 sps. The dense sample rate of 600 and decimation factor of 3 are used in routine processing, but different values for these two parameters may be provided for non-standard processing.

With the frequency domain method, equal-length segments of densely interpolated time series data are transformed to the frequency domain with a Fast Fourier Transform. Transformed values are instrument corrected, filtered, then transformed back to the time domain. The separately filtered segments of data are fitted back together using an "overlap-add" method similar to that described in reference [9].

Some analog records require that an operator manually digitize a sharp peak and its approaches where the trace is too faint for the laser scanner to follow. The operator does not digitize as densely as the automatic scanner can and cannot digitize where the trace is too faint to follow. Sparse digitization may result in a time series having sharper peaks containing larger high frequency Fourier components than the series would have had if the trace had been more accurately digitized. Since high frequency components are amplified by the instrument correction, records for which these problems are severe may need to be processed without instrument correction or with a high-cut filter having a relatively low roll-off frequency.

The results from HIFRIC may be plotted with the TSPLOT program.

6.2 Standard HIFRIC process:

1) Calculate weights for the three convolution operators that will be used in step 4. Each operator has 61 weights which will span 0.1 second of 600 sps data.
2) Read input data that have been prepared by the SCALE program or its equivalent. These data, y(t), represent measured, scaled instrument response and are equal to $-\omega^2 x(t)$, where x(t) is the displacement of the mass relative to the instrument case and ω is the natural frequency of the instrument.
3) Interpolate linearly to 600 sps.

transition band providing a cosine taper from full to zero frequency response. The remaining bands, from 100 to 300 Hz, are assigned zero frequency response. After filtering, the data are decimated by removing 2 out of every 3 points, reducing the sampling density from 600 sps to 200 sps. The dense sample rate of 600 and decimation factor of 3 are used in routine processing, but different values for these two parameters may be provided for non-standard processing.

With the frequency domain method, equal-length segments of densely interpolated time series data are transformed to the frequency domain with a Fast Fourier Transform. Transformed values are instrument corrected, filtered, then transformed back to the time domain. The separately filtered segments of data are fitted back together using an "overlap-add" method similar to that described in reference [9].

Some analog records require that an operator manually digitize a sharp peak and its approaches where the trace is too faint for the laser scanner to follow. The operator does not digitize as densely as the automatic scanner can, so this portion of the time series may appear sharper and may contain higher frequency Fourier components than it would have if it had been more accurately digitized. Since high frequency components, even those falling in the lower portion of the transtition band, are amplified by the instrument correction, records containing serious inaccuracies of manual digitization should be processed either without instrument correction or with a high-cut filter having a relatively low roll-off frequency.

The results from HIFRIC may be plotted with the TSPLOT program.

6.2 Standard HIFRIC process:

1) Calculate weights for the three convolution operators that will be used in step 4. Each operator has 61 weights which will span 0.1 second of 600 sps data.
2) Read input data that have been prepared by the SCALE program or its equivalent. These data, y(t), represent measured, scaled instrument response and are equal to $-\omega^2 x(t)$, where x(t) is the displacement of the mass relative to the instrument case and ω is the natural frequency of the instrument.
3) Interpolate linearly to 600 sps.

4) Apply convolution operators centered at every third sample of the 600 sps interpolated input data to compute z, z', and z" at 200 sps, where z is a high-cut filtered version of the input data, y. Combine z and its two derivatives according to the damped harmonic oscillator equation, where

$$\text{corrected acceleration} = z + z'2n/\omega + z''/\omega^2$$

n is the fraction of critical damping of the instrument, and
ω is the natural frequency of the instrument in radians per second.

This 4th step simultaneously provides:
- instrument correction;
- high-cut filter with pass band from 0 to 50 Hz., cosine taper between 50 and 100 Hz., and rejection band from 100 to 300 Hz.;
- decimation to 200 sps.

The high-cut filter is applied to prevent aliasing when the data are decimated to 200 sps.

6.3 Running HIFRIC

To run HIFRIC, type:

```
$ HIFRIC <output file>,<msgfile>=[SCALE file], -
        [period],[damping],<sps>,<ndense>, -
        <non-standard flag>,<fr>,<ft>
```

<Output file> is a name for the output file. If this is not given on the command line, the default output file name will have the same name as the input file except for the first character in the suffix, which will be a 'G'.

<Msgfile> is the name for an optional disk file that will receive the run messages and diagnostics that would normally go to an interactive user's terminal or to a batch job's log file.

[SCALE file] is the name of the input file, a file prepared by the SCALE program.

[Period] is the period of the recording transducer, in seconds.

[Damping] is the damping of the recording transducer, as a fraction of critical damping.

[Period] and [damping] are required on the command line only if the period and damping values in the header block in the input file are incorrect or missing.

<Sps> is the intended sample rate for the output data in samples per second. Default =200.

<Ndense> is the ratio of the dense sample rate to the final sample rate, <sps>. <ndense> must be an integer. Default=3. (The instrument correction and filter are applied to the more densely sampled data.)

The optional non-standard flags are "INTERP" and "FDIC". With INTERP on the command line, HIFRIC will interpolate, but not instrument correct, in which case the [period] and [damping] values need not be given. With FDIC on the command line, HIFRIC will use the alternative frequency-domain instrument correcting method rather than the standard time-domain method.

<Fr> to <ft> is the transition band, in Hertz, for the transition taper in the filter used with the FDIC option. Default = 50. to 100. Note that with the standard option, the transition band is determined not from these two input parameters but from the dense sample rate used (dsps/12 through dsps/6.)

The meaning of the numeric parameters ([period],[damping], <sps>,<ndense>,<fr> and <ft>) depend on the order in which they are given, so give a null value (two consecutive commas) if you wish to use a default value among other numeric parameters you wish to set.

Examples --
```
   $ HIFRIC TEMPA.G01=EDA.R01, .040,.570
   $ HIFRIC TEMPB.I01=EDA.R01, INTERP
   $ HIFRIC TEMPC.G01=EDA.R01, .040,.570,,,FDIC,40.,90.
```

6.4 HIFRIC Limitations and Possible Future Modifications

The subroutines (SETWTS and FDIC) that apply the two alternative correction algorithms can be modified to alter characteristics like the number of convolution weights, the overlap and segment size, and so forth. Some of the choices for tuning these algorithms should be run options rather than predetermined code in order to handle atypical data. More investigation needs to be done with both instrument correcting subroutines before we understand what their limitations are when used with atypical data or with non-standard filter parameters, and with which situations, if any, one subroutine is more appropriate than the other.

The time-domain subroutine (SETWTS) imposes more restrictions on its input parameters than does the frequency-domain subroutine (FDIC). These restrictions aren't inherent in the method, but stem from the way it was coded. The restrictions are:

- The width of the filter's transition band must be the same as

the width of the pass band. If the program is to allow a transition band smaller than the pass band, more weights may be needed for the convolution operators than the program now provides. With more weights, the program will run more slowly.
- The band width (cycles per second) must divide evenly into the sampling rate (samples per second).
- The number of bands (6 in standard processing) must be an even multiple of two if decimation is performed, and the ratio of dense to decimated data must equal bands/2.

As is the case with the time-domain subroutine, the range of parameters for which the frequency-domain subroutine is appropriate has not been investigated carefully yet. The lengths of the separate segments of data and their overlapping area are now set by the program to be 1024 points and 256 points respectively. These lengths may not be suitable for atypical data or atypical filter bands. Until the code is changed to allow the user to specify the segment and overlap lengths, users should beware of results unless run parameters are such that:

- The frequency at which the filter's transition taper begins is less than or equal to half the frequency at which the taper ends.
- Frequency at the end of the taper is less than half the sampling rate.
- Instrument period and damping are values typical of SMA accelerographs (period between 0.04 and 0.1 second; damping about 0.6 of critical damping). Sampling rate between 50 and 1000 samples per second.

Both instrument correcting algorithms assume that the time series is zero before the first sample of actual data. Perhaps leading points should be set to a cosine taper to zero. Or perhaps the first few points should be dropped from the output (Michael Raugh, the author of the time-domain algorithm, did delete NWTS points from the end of the series.)

The convolutions done in subroutine OPRTR should accumulate positive and negative sums separately, then add the two sums together at the end (according to Michael Raugh, the author of OPRTR.)

If a DR100 file is given as input, HIFRIC should warn the user that the equation used for instrument correction is not appropriate. If instrument correction is required for digital recorded data, we shall need new options in HIFRIC or new programs to apply corrections appropriate to the data.

The INTERP option simply interpolates linearly between the data samples given; it does <u>not</u> apply a high-cut filter to prevent the aliasing effect of reducing the sampling rate. Another option could be added to the program that would provide high-cut filtered, interpolated data that has not been instrument corrected.

CHAPTER 7

CORAVD
Corrected Acceleration Velocity and Displacement Calculating Program

7.1 Overview

CORAVD calculates velocity and displacement from acceleration and optionally performs baseline correction of two possible types. One option makes a linear correction to the velocity and another option filters long periods from acceleration and velocity.

Triggered analog records do not begin instantaneously with the beginning of the earthquake motion, but only after the motion has become strong enough to trigger the recorder. The linear baseline correction option in CORAVD establishes a reasonable value for the acceleration and velocity at the beginning of such a record provided that some portion (or all) of the first approximation of velocity has a reasonably linear trend. A fitted line is subtracted from the velocity and a constant, equal to the slope of the line, is subtracted from the acceleration. The portion of the velocity trace to be fitted is specified as a run parameter. No attempt is made to estimate an initial value for the displacement. The displacement is calculated from the velocity after the velocity has been corrected.

Digital recorders with pre-event memory show, in their records, several seconds of motion that occurred before triggering. Initial values of zero for acceleration, velocity and displacement are appropriate for these records (provided the recording system operated as designed) and the linear correction is generally not required.

The linear correction gives good results for some accurately digitized records, but many records will require that long periods be filtered from the data before reasonable displacements can be calculated or before reasonable response spectra can be calculated in the RSPECT or PHASE3 programs. For routine processing a bidirectional Butterworth filter is used, and several other filter algorithms are available for use in special studies.

By default, CORAVD does no filtering and applies the linear baseline correction, fitting the entire length of the velocity time series. Processing with no low-cut filtering is appropriate only for high-quality, accurately digitized records, so users must reconsider for each particular record whether or not the default processing is appropriate.

The results from CORAVD may be plotted with the TSPLOT program.

7.2 Standard CORAVD process:

1) Integrate acceleration to compute a first approximation of velocity. Use trapezoidal integration and use zero as the initial velocity.
2) Subtract the linear least squares fit of the velocity from the velocity to establish an initial value for the velocity. Fit the entire velocity time-series unless user has specified that just a portion of the velocity, maybe just the quiet portion at the end of the time-series, be used.
3) Subtract a constant from the acceleration, the constant being the slope of the line fitted to velocity.
4) Filter low frequencies from velocity and acceleration, if requested. Use the same filter parameters for both.
5) Integrate velocity to compute displacement, using zero as the initial displacement.

7.3 Running CORAVD

To use CORAVD, type:

```
   $ CORAVD <output files>,<msgfile>=[input file], -
            <begin fit>,<end fit>,<taper fit>, -
            <filter type>,<filter parameters>, -
            <filter type>,<filter parameters>
```

Examples:

```
   $ CORAVD ATEMP.=[AGRAM.TESTDATA]EDA.G03, 6.0,40.0

   $ CORAVD BTEMP.=[AGRAM.TESTDATA]EDA.G03,    0,    0, BI,0.17,2
```

<Output files> =
optional names for the 3 output files. If these aren't given on the command line, the default output file names will have the same name as the input file except for the first character in the suffixes, which will be 'A', 'V', and 'D' to indicate that the files contain acceleration, velocity, and displacement.

<Msgfile> =
name of an optional disk file to contain run messages and diagnostics that would normally go to an interactive user's terminal or to a batch job's log file.

[Input file] =
name of the input file. This file should be an output file from the HIFRIC or DR100 programs.

<Begin fit> and <end fit>
specify the endpoints, in seconds, of the least squares fit line to be used for the linear base line correction. If these are not given, the entire velocity trace will be fitted. If they are both given as zero, as is recommended when processing data that requires filtering, then no linear correction will be performed.

<Taper fit> specifies the fraction of the least-squares fit range in which to apply a cosine tapered weighting factor. The taper is applied to both ends of the fit range. <taper fit> must be between 0.0 and 0.5. Default value =0.0.

<Filter type> =
"BI", "UNI", "FFT", or "ORM" for bidirectional Butterworth filter, unidirectional Butterworth filter, FFT filter, or Ormbsy filter. It is the BI filter that is used for routine processing of data that requires filtering. The other filters are provided only for special studies.

If two sets of filter types and filter parameters are given, the first is for a low cut (or high pass) filter and the second is for a high cut (or low pass) filter. The ability to apply a high cut filter in CORAVD is intended just for special studies in which the high cut filter that is normally applied in HIFRIC has been bypassed. Beware against refiltering high frequencies if they have already been filtered in HIFIRC though; that would distort the data unnecessarily. The BI and UNI filters have not been coded to allow high cut filtering yet; it is only the FFT and ORM filters that can be used for such.

<Filter parameters>.
Filter parameters are somewhat different for each type of filter. They specify the transition band between the pass band and the rejection band of the filter. (Although other filter tuning parameters may be added later.)

For the FFT and Ormsby filters, the transition is specified by giving a roll-off frequency, <tfreq>, at which the filter shall start to roll off and a transition bandwidth, <tband>, in which the transfer function of the filter shall decrease from one to zero. The transition is a cosine half-bell taper in the FFT filter. In the ORM filter, the transition is a linear ramp with a short parabolic section on both ends where the ramp meets the pass band on one end and the rejection band on the other end.

For the Butterworth filters, the transition is given as a roll-off parameter <m> (which is equal to half the order of the Butterworth filter in conventional terminology) and a corner frequency, <tfreq>. Unlike the <tfreq> given for FFT and ORM filters that specifies the <u>begining</u> of the transition band, <tfreq> for the BI filter is the frequency at which the transfer function of the filter is down to 1/2 and <tfreq> for the UNI filter is the frequency at which the transfer function is down to $1/\sqrt{2}$. The BI filter is simply the UNI filter applied twice, once in a forward and once in a backward direction. It is used to eliminate the phase distortion in the UNI filter and has a transfer function that is the square of the transfer function for the UNI filter. The transfer functions, T_{bi} and T_{uni}, are given by

$$T_{bi}(if) = T^2_{uni}(if) = \frac{1}{1 + (\langle tfreq\rangle / f)^{4\langle m\rangle}}$$

where the independent variable, f, is frequency.

For a BI directional Butterworth filter, give
    BI,<tfreq>,<m>
where
    <tfreq> = corner frequency in Hertz. Default=0.1.
    <m> = rolloff parameter, an integer. Default=2.

For a UNI directional butterworth filter, give
    UNI,<tfreq>,<m>

For an FFT filter, give
    FFT,<tfreq>,<tband>
where
    <tband> = width of the transition band, in Hertz.
    Default=0.1.

For an ORMsby filter, give
    ORM,<tfreq>,<tband>,<ormk>
where
    <ormk> = a fraction of the rolloff ramp width in which to apply parabolic smoothing. Each end of the linear rolloff ramp is tapered with a parabola to avoid discontinuities where the ramp meets the pass band and the rejection band.

    Default value for <ormk> =0.1.

7.4 CORAVD Limitations and Possible Future Modifications

It is not recommended that the linear baseline correction be used when a filter is used. Although CORAVD now applies the linear correction by default even if a filter is requested, it might be less confusing if user were required to request the linear correction explicitly whenever it is to be applied.

In order to avoid a wrap-around effect from the filters, the actual data is padded with trailing zeros before filtering. (Except when using the UNI filter which doesn't require padding.) At present, the FFT filter pads with enough trailing zeros to bring the number of samples out to the nearest power of 2. The BI filter pads with half as many trailing zeros as there are samples in the actual time series. Both these pad lengths may be too short in some situations. Although such has not been incorporated into CORAVD yet, William Joyner and Peter Mork (both USGS) performed some tests indicating that an appropriate pad length for a Butterworth filter is 1.5mT, where m is the roll-off parameter for the Butterworth filter and T is the reciprocal of the corner frequency.

The zero padding will create a sharp step in a time series if the series does not begin and end at zero. This step may cause problems if the high-cut filter option is ever used. This same problem needs to be considered with respect to the high-cut filter in HIFRIC too. A tapering option like the DATATAPER option used in the FASPLT program has been coded in CORAVD but has not been tested yet. To try it, give DATATAPER among the filter options. William Joyner suggests that a suitable width for such a taper would be 1.0/(roll-off frequency of the high-cut filter).

The ORM filter would be faster for low cut filters if we provided an option for smoothing and decimating the baseline before it was filtered as was done in the PHASE2 program.

CHAPTER 8

FASPLT program

8.1 Overview

FASPLT plots the Fourier amplitude spectrum of evenly-sampled time series data from CORAVD, HIFRIC, or DR100 output files. FASPLT is intended primarily for processing base-line corrected CORAVD acceleration files, but it may also be used with files from the earlier stages of processing to illustrate the frequency content of unfiltered data. The input data must be evenly-sampled, however. If you wish to use FASPLT with SCALE results, which are not equally-spaced, you must first use HIFRIC with the INTERP option to create a file of uncorrected but interpolated data.

The Fast Fourier Transform (FFT) method is used to calculate the Fourier spectrum. Before transforming from time domain to frequency domain, the time series is padded with trailing zeros to bring the number of data points to a power of two, as required by the FFT algorithm used in FASPLT.

In transforming the time series from the time domain to the frequency domain, the FFT returns half as many, plus one, complex values as there were real-valued ordinates in the time series. The frequency range of the new sequence runs from zero Hz to the Nyquist or "folding" frequency of $1/(2\Delta t)$ Hz, where Δt is the sampling interval of the original time series, in seconds. The interval between samples in the new sequence is 1/T Hz, where T is the total time spanned by the padded time series ($= 2^n \Delta t$). The spectra returned by the FFT for different time series having the same sampling interval (Δt) will have the same range of frequencies but the sampling interval in the frequency domain will be different if the padded time series have different lengths. By default, FASPLT pads the time series with trailing zeros to bring the number of samples to the nearest power of two. The NSAMPLES option allows the user to specify which power of two to use as the padded (or truncated) time series length. For example, FASPLT would, by default, pad a 4-second time series using the standard 200 sps sampling rate to 5.12 seconds, or 1024 samples. FASPLT would pad a 30-second time series using the same 200 sps sampling rate to 40.96 seconds or 8192 samples. The sampling interval in the frequency spectrum of the 1024-sample time series is 0.195 Hz and of the

8192-sample time series is 0.024 Hz. If a user wished to compare two such spectra point-by-point, they should use the NSAMPLES option to request FASPLT to pad the shorter time series out to the same number of samples as was used for the longer time series.

If the first or last point of the time series is significantly different from zero, there will be a steep step in the padded time series where the actual data meets the padded area. This sharp step will introduce spurious frequencies ("leakage") into the series. Although CORAVD data has been "baseline corrected" so the ending value of the results from CORAVD should be close to zero, the ending value may not be precisely zero. The beginning value may not be zero either, since the linear baseline correction in CORAVD may have established a non-zero beginning value for the series. To minimize the effect of such a step between the actual time series and the zero pad area, the step may be tapered with one of several options provided by the program. The options are "ZCROSS", "DATATAPER", and "NOTAPER". With the default "ZCROSS" option, values for points before the first zero crossing and values after the last zero crossing are all reset to zero. With the "DATATAPER" option, a section of each end of the time series is multiplied by a cosine half-bell taper to bring the value at the first and last point in the series to zero. The length of each tapered section is, by default, 0.1 times the length of the unpadded time series. The user may specify a different factor than 0.1 for determining the taper length, however. With the "NOTAPER" option, no tapering is performed.

After the real-valued time series has been transformed to a complex-valued Fourier spectrum, scalar Fourier amplitude values are calculated from the complex Fourier transform at each frequency step. By default, the resulting amplitudes are plotted without further processing, but they often show such dense fluctuations that the general shape of the curve is obscured. By option, the squared amplitudes may be smoothed with a running mean before they are plotted.

Options in the program also provide for drawing plots of the Fourier amplitude spectra of integrals and/or derivatives with respect to time of the input time series. Each amplitude value in the spectrum of the integral is calculated by dividing the corresponding amplitude of the original function by the corresponding angular frequency, ω; the amplitude of the derivative is calculated by multiplying by ω. This frequency-domain method of integration used in FASPLT may be compared to the time-domain method used in CORAVD by comparing a Fourier ampltude spectrum plot of a CORAVD displacement file to a Fourier ampltude divided by ω^2 spectrum plot of the CORAVD acceleration file from which the displacement was calculated.

8.2 FASPLT processing steps

- Read the input time series. By default, the entire time series is used, but user may specify that just a subinterval be used.

- Pad the input data with trailing zeros. By default, the number of points in the padded time series is the next power of two greater than the number of actual data points.

- Apply tapering option.

- Apply FFT to transform the padded time series from time-domain to frequency-domain.

- Calculate squared Fourier amplitudes from the complex Fourier spectra resulting from the FFT. At each frequency in the sequence, $\text{amplitude}^2 = I^2 + R^2$, where R is the real component and I the imaginary component of the complex Fourier spectrum at the frequency under consideration.

- If requested by the "SMOOTH" option, smooth the squared amplitude values with a weighted running mean. The weighting function has a triangular shape and, in the default case, has three terms: 1/4, 1/2, and 1/4. The end points of the series are smoothed as though there were leading and trailing zeros on either side of the actual sequence of squared amplitude values.

- Take the square root of the squared amplitude values.

- If requested by the "FPS" option, calculate Fourier phase at each frequency in the sequence too. Phase = arc tangent (I/R).

- If the "FAS" option (the default) was requested, plot amplitude with respect to frequency.

- If the "FPS" option was requested, plot phase with respect to frequency.

- If the "FASI" option was requested, divide each smoothed amplitude value by its corresponding angular frequency, ω. Angular frequency in radians per second = 2π times the frequency in Hz. Plot the amplitude/ω spectrum.

- If the "FASI2", "FASD", or "FASD2" options were requested, repeat the last step but multiply by $1/(\omega^2)$, ω, or ω^2 rather than dividing by ω.

- If two input files were given, repeat all of the above steps for the contents of the second input file. Then, for each curve that was plotted (FAS, FASI, FASI2, FASD, FASD2 and/or FPS) take the ratio of the first file's curve to the second file's curve and plot that ratio.

8.3 Running FASPLT

There are four different versions of FASPLT: SFAS plots on CRT screens, VFAS on the Versatec plotter, CFAS on the Calcomp plotter, and IFAS plots through the device-independent VIEWER software available at NSMDC. To run FASPLT, select the name of the version you wish (SFAS in these examples); then, for the simplest form, use —

```
$ SFAS  [input file]
```

Or, to override the default processing parameters, use —

```
$ SFAS  <msgfile>=[input file1],<input file2>, -
         <tbeg>,<tend>,<"TSPLOT">,<"NONOISE">, -
         <computing options>,<plot axis options>, -
         <plot title options>
```

<Msgfile> is the name of an optional output file that will receive the run messages and diagnostics that are normally displayed at the user's terminal. It is often useful to specify a <usermsg> file when using SFAS, since run messages displayed on the screen can be obscured by the plots. If <usermsg> is not given, the equal sign is not needed.

[Input file 1] is the name of the input file. The input file must be a Vinton-Fletcher style blocked-binary file containing evenly-sampled data. The file may contain acceleration, velocity or displacement, but for standard processing it is acceleration data from CORAVD that is given to FASPLT.

<Input file 2> is the name of a second input file. This is used when plots of spectral ratios are required.

<Tbeg>,<tend> are optional real numbers representing the times, in seconds, that bracket the section of the input time series to be processed. By default, <tbeg>=0.0 and <tend> is the time corresponding to the last point on the input file.

TSPLOT is an optional keyword that indicates that a plot of the padded, tapered time series should be shown above each spectrum plot.

NONOISE is another optional keyword that indicates that the spectra plots should not show amplitudes at frequencies above the transition frequency used with the high-cut filter applied in HIFRIC or below the transition frequency used with the low-cut filter applied in CORAVD.

<Computing options> are optional and may include

1) The keyword "NSAMPLES" followed by an integer number in parenthesis. The number must be a power of two and it specifies the length of the padded or truncated time series to be transformed by the FFT.

2) One of the tapering option keywords "DATATAPER", "PADTAPER", "ZCROSS" or "NOTAPER". The "DATATAPER" and "PADTAPER" keywords may be followed by a real number in parenthesis. The number must be between 0.0 and 0.5 and specifies the length of the tapers as a fraction of the unpadded time series length.

3) One of the smoothing option keywords "SMOOTH" or "NOSMOOTH". The "SMOOTH" keyword may be followed by an integer number in parenthesis. The number must be odd and it specifies the number of weights to use in the triangular smoothing function. Note that SMOOTH (1) and NOSMOOTH are equivalent.

4) Any or all of the keywords "FAS", "FASI", FASI2", "FASD", "FASD2" and "FPS". These keywords indicate the type of data to be plotted: Fourier amplitude spectrum of the input time series (FAS), Fourier amplitude spectrum of the time series integrated with respect to time (FASI), integrated twice (FASI2), differentiated (FASD), differentiated twice (FASD2), or Fourier phase spectrum (FPS).

The default computing options are DATATAPER(0.1), NOSMOOTH, FAS and NSAMPLES(<n>), where <n> is the nearest power of two greater than the number of samples given in the input data.

<Plot axis options> are optional and may include any or all of the four keywords that indicate the type of axes to use in the spectra plots. The keywords are LOGLOG, LOGLIN, LINLOG and LINLIN. They indicate whether the x and y axes should be plotted in logarithmic or linear scale. Note that it is only in LINLIN and LINLOG plots that the first point, that for zero frequency, will be plotted. The LOGLOG and LOGLIN plots omit the zero frequency point.

By default, the program will choose the range for each axis, but users may specify these too, if they wish. Each type of axis keyword may be followed with parenthesis containing four numbers, the numbers separated from one another with commas or blanks. The numbers indicate the beginning value for the x-axis, ending value for the x-axis, beginning value for the y-axis and ending value for the y-axis, respectively. A dash (-) may be used in place of any of the four numbers to indicate that the program should choose an appropriate number. For linear axes, the number is the actual value to use at one end of an axis. For log axes, the number is an exponent of ten to be used at one end of an axis. The user-specified axis ranges will be adjusted, if necessary, so that log axes will begin and end at even powers of ten, and linear axes will begin and end at some convenient number close to the user-specified number.

If user has requested that several types of data be plotted (more than one of the FAS, FASI, FASI2 ... options) and also wishes to specify a different axis range for each type of plot, he must repeat the plot axis options for each type of plotted data option. To do so, follow each type of data option with the axis options applicable to that data. Example:
FAS, LOGLOG (-3,-,-3,4), FASI, LOGLOG(-3,-,-2,3).

The default plot axis option is LOGLOG(-,-,-,-).

<plot title options> are optional and may include one of the keywords "TOP", "FIG", "DOC", "NOLABELS", "NOTITLES" or "AXESONLY"; and either the name of a text file that contains one or more lines of text for the plot title, or a character string containing titling text. The character string, if given, must be delimited by double quote marks (") and is used only with the "FIG" and "DOC" options.

The TOP keyword (the default) indicates that the plot title should be shown at the top of the page.

The FIG and DOC keywords provide titles in a form suitable for publication in the strong-motion data reports issued by the USGS. With FIG, the plot title will be shown at the bottom of the plot page, as though it were a figure caption. With DOC, the title will be centered at the top of the page. With FIG and DOC, titling information is, by default, retrieved from the header blocks in the data file. Since the information may not be in the header blocks, user may give the primary title line on the FASPLT command line. The title character string, if given, must be enclosed in double quotes ("..title..") and only one or two title strings may be given, one for each data file.

There are four fields in the FIG/DOC primary title line. The fields are delimited with semicolons and two adjacent semicolons in a title string given on the FASPLT command line indicate that FASPLT should fill out the field using information from the header blocks in the data file. The first field contains a station and event identification, the second field the orientation of the recorded movement, the third field the date of the recording, and the fourth field the time. Example:

"Coalinga, Anticline Ridge; 360 deg.; May 9,1983; 02496 UTC."

If user wished to use the station and orientation information from the header in the data file, but provide the date and time on the FASPLT command line, he would use the following as a title string.
";; May 9,1983; 02496 UTC."

The FIG keyword may be followed by a figure identifier in parenthesis, as in: FIG (A.3). This identifier will be shown on the plot above the figure caption. If the identifier is an integer, or ends with a dot followed by an integer (e.g.: A.3), and more than one plot is produced in the FASPLT run, then that integer will be incremented by one for each plot.

The NOLABELS, NOTITLES and AXESONLY keywords are intended for use with IFAS and the plot overlay capability provided by the VIEWER package at NSMDC. With NOLABELS, no labels or axes will be plotted, just the curve. AXESONLY is like NOLABELS except that the axes are plotted as well as the curves. With NOTITLES, the axes will be plotted and labeled, but there will be no title at the top of the plot.

The default plot title options are TOP with the name of the input file as the plot title.

Sample FASPLT commands:

```
$ SFAS  PUB1:[AGRAM.TESTDATA]EDA.R01

$ SFAS  FROMSFAS.TMP =PUB1:[AGRAM.TESTDATA]EDA.R01, -
        ZCROSS,SMOOTH, -
        FAS,FASI,LINLIN,LOGLIN,LOGLOG, -
        FIG, 'THIS IS ONE LINE OF PLOT TITLE'
```

CHAPTER 9

PHASE3 and RSPECT
Response Spectra Programs

9.1 Overview

The PHASE3 program calculates response spectra and Fourier amplitude spectra. The relative velocity response spectrum and the Fourier amplitude spectrum appear in one plot with linear axes. The pseudo-velocity response spectrum appears in another plot with tripartite log-log axes. The Fourier spectrum shown in the first plot is calculated (from its definition) at the same frequencies used in the response spectrum calculations, unlike the Fourier amplitude spectrum shown in FASPLT plots, which is calculated at equal intervals determined from the number of samples given in the input time series.

Unlike the other programs in the AGRAM series, PHASE3 has not been reorganized recently. It has often been modified to adapt the code to different computers and different user requirements. The resulting code is cumbersome, awkward to use, and unnecessarily slow in execution. A new program, RSPECT, will eventually replace PHASE3, but as yet it is only available for testing, not for production use. Use the on-line help facility by typing "$HELP RSPECT" to learn the current status of this new program.

Other response analysis programs, such as those used to perform the calculations in references [17], [18] and [19] may also eventually be reorganized and added to the AGRAM series.

9.2 Running PHASE3

PHASE3 requires four input files. They must all have the same prefix and their suffixes must be .A01, .A02, .A03 and .TIT. The first three files are corrected acceleration files produced by the CORAVD program and the fourth file is a text file containing the text to be shown at the top of the plots. The three data files are usually those for three components recorded together, but if fewer than three data files are to be processed, enough copies of one of the actual data files or of a dummy data file must be made to satisfy PHASE3's requirement for three input data files. The text file must contain three lines of title information. For preliminary runs, the title file may simply contain three blank lines.

PHASE3 is really two programs: PHASE3 does the calulations and PH3PLT plots the results. PHASE3 issues two prompts. Answer the first with the prefix of the four input files, answer the second with the name of the output file. PH3PLT issues one prompt; answer with the name of the output file generated in PHASE3.

9.3 Running RSPECT

To run RSPECT, type:

```
$ RSPECT  [input file]
```

[input file] is the name of an acceleration file produced by CORAVD.

CHAPTER 10

Support Programs

10.1 Overview

The support programs described in this chapter manipulate blocked binary data files from the various stages in the AGRAM processing. They perform plotting, display, reformatting and other miscellaneous functions. The TSPLOT, BBFILE, BWRITE, REFORM, ROTATE and BBDATA programs are available now and more support programs may be added in the future.

10.2 Plotting Programs and Plotting Media at NSMDC

At NSMDC, plots are made with CRT screens, with a Versatec electrostatic plotter, and with a calcomp pen plotter. There are several different versions of each of the plotting programs in the AGRAM series: one for each type of plotting device, and one for the VIEWER device-independent plotting package in use at NSMDC. The different versions of the same program are distinguished from one another by the first character in the name used to invoke the program. STSP, VTSP, CTSP and ITSP are the several names that can be used to invoke the TSPLOT program. It can be invoked using the name TSPLOT too, but that is equivalent to ITSP. STSP plots on CRT screens, VTSP plots on the Versatec plotter, CTSP plots on the Calcomp plotter, and ITSP plots through the VIEWER package. There are several different versions of the FASPLT program, likewise: SFAS, VFAS, CFAS, and IFAS. Other plotting programs that may be added to the AGRAM series in the future will be named similarly.

ITSP and the VIEWER programs allow users to display a TSPLOT plot on a screen, then to choose whether to reproduce the same plot on a hard-copy device. ITSP and the VIEWER programs also allow users to combine several plot pages into a new display. ITSP is run just like the other versions of TSPLOT except that the user must respond to prompts asking for VIEWER operating modes. ITSP will generate a disk file named BATCH.PLT that VIEWER will subsequently need to access. After running ITSP, run one of the VIEWERs: VIEWER for screen plots and plot file manipulation, or VIEWERR for Versatec plots. There is no Calcomp version of VIEWER on either the VAX 11-750 or the VAX 11-780 yet. For Calcomp plots, transfer the BATCH.PLT file from the VAX to the RSX machine and run LB0:[7,11]VIEWERC. Once

VIEWER begins to prompt you with "viewer>", type "help" to learn how to proceed. If you need more help with the VIEWERs, use the on-line help facility by typing "$ HELP LIBRARIES /PLOT".

The STSP, VTSP and CTSP "programs" are, at present, not really separate programs (as they were when installed on the RSX machine) but are merely command decks set up to run ITSP and VIEWER with their results sent to the appropriate device. As a result, a lot of unneccessary run messages are generated. STSP, set up this way, does not wait for you to type the return key to indicate when you are through viewing a plot (as it did on the RSX); it simply waits 8 seconds and then continues processing. If you wish to see your screen plot for longer than 8 seconds, use the no-scroll key.

CTSP, at present, does nothing but tell you that the Calcomp plotter is not connected to the VAX yet! The calcomp plotter may be attached to the VAX 11/750 in the future; at that time, the version of CTSP on that VAX will be changed to provide Calcomp plots. There must be two pens in the pen holder when making Calcomp plots with any of the AGRAM plotting programs. The first (right-most) pen is used for the curves and axis lines, the second pen is used for characters. Characters plotted with a felt tip pen make better ZEROX copies than those done with a ball point pen.

A new plotting package (DI3000) may replace the VIEWER package at NSMDC in the future. The STSP, VTST, CTSP, and ITSP plotting "programs" will then be adjusted to use the new package.

10.3 Time Series Plotter, TSPLOT

TSPLOT plots time series data from DR100, SCALE, HIFRIC, or CORAVD. It cannot plot IOMTAP or BUTTER output files yet.

As previously mentioned, There are four different versions of TSPLOT: STSP plots on CRT screens, VTSP on the Versatec, CTSP on the Calcomp, and ITSP plots through the device-independent VIEWER software available at NSMDC.

To run TSPLOT, type the name of the version you wish, followed by a list of file names. The file names specify the data files you wish to have plotted. You may also include the name of a text file that contains a top-of-plot title.

Example:

```
$ STSP [AGRAM.TESTDATA]EDA.A01,.V01,.D01,[]TOPPLOT.TTL
```

The example above uses default scaling and labeling options. Each plot page will show 20 seconds of each curve (one curve for each file given on the command line). The curves are shown in separate strips across the page, each strip with its own y-axis, and the width of the strips (length of the y-axes) depending on the number of curves on the page. The first two significant digits, plus one, of the peak value of a curve are used for the scale on that curve's y-axis. The title at the top of the plot will come from the text file specified in the command line, or if there was no text file, the title will consist of the names of the data files.

To alter the size of the axes, you may include up to five numeric parameters after the file names on the command line. The meaning of these numbers depends on the order in which they are given, so give a null value (two consecutive commas) if you wish to use a default value among other parameters you wish to set. The numeric parameters and the order in which they must be given are: <tbegin>,<tend>,<spp>,<ysize>,<yspace>.

<tbegin>	is the time at which the plot will begin. Default=0.0.
<tend>	is the time at which the plot will end. Default= ending time of the longest curve.
<spp>	are the number of seconds to appear across each plot page. Default =20.0. If <tend> - <tbegin> is greater than <spp>, more than one page will be plotted.
<ysize>	is the size of the y-axes, given as a fraction of the plot page.
<space>	is the size of the space to be left between the plot strips, given as a fraction of the plot page.

To alter the vertical scale for any curve, you may include the maximum range for that curve's y-axis in parentheses after the file name. The y-axis will range from minus to plus the value in parentheses. If you wish to use the peak value of the curve, include the letter "p" in the parentheses rather than a number.

To alter the way peaks are labeled, include the keyword "ARROW" or "NOPEAK" in the command line. ARROW will plot a little arrow that points to the peak, and NOPEAK will not put any label at all next to the peak.

To alter the other labels, include one of the following keywords in the command line: "SEB", "NOLABELS", or "AXESONLY". With SEB, the top of plot titles will appear in the format used in the data reports published by the USGS. NOLABELS and AXESONLY are intended for use with ITSP and the VIEWER device-independent plotting package. With NOLABELS, no labels will be plotted, except perhaps those next to the peaks. AXESONLY is like NOLABELS except that the axes are plotted along with the curves.

More examples:

```
    $ STSP MYDATA.R01(500.),.R02(300.),.R03(500.),TOPPLOT.TTL, -
         SEB, 3.0,10.0,7.0
    $ STSP MYDATA.R01(P),.R02(P),.R03(P),,,,0.18
```

To compare the shapes of curves plotted in several different TSPLOT runs, it is a good idea to provide a value for <ysize> so that all the curves to be compared use a y-axis of the same size. If <ysize> is not specified, TSPLOT will choose a value that depends on the number of lines in the top-of-plot title and on the number of curves to be plotted on the page. <Ysize> values of 0.7, 0.3, 0.2, and 0.14 work well for one, two, three, and four curves per plot page, respectively. To be more precise in choosing a value for <ysize>, divide up the vertical plotting space by the number of curves to be plotted and subtract a bit (0.04) for space between the curves. The vertical space, as a fraction of the plot page, left for plotting after labels and margins are allowed is (0.85 - 0.01*(3 + number of top-of-plot title lines)).

TSPLOT Limitations and Possible Future Modifications:

The "SEB" keyword will be replaced with "FIG" and "DOC" keywords like those used in the FASPLT program. A "NOTITLES" keyword like that in FASPLT will also be added to TSPLOT.

TSPLOT could allow the portion of the page to be left for margins to be specified at run time. Allow four more numeric run parameters to represent the left, right, top and bottom margins in terms of fractions of the plot page. All the code for this is in place except for command line interpretation.

TSPLOT could offer an option to specify the plot scale in centimeters rather than in fractions of the plot page so we could reproduce a trace in the same scale as the original (or an enlarged original). Most of the code for this is in place except for command line interpretation. Maybe if <ysize> is given greater than 1.0, TSPLOT should interpret the <ysize> value as the y-axis length in centimeters and interpret the <spp> value as the length in centimeters for each second along the x-axis.

Some users have asked for more control over what does and does not get labeled with options like NOLABELS and AXESONLY.

A means of plotting the results from IOMTAP and from BUTTER needs to be added to the AGRAM series. These plotting functions may be added to TSPLOT, or they may be provided by separate, new, programs: IOMPLT and BUTPLT.

10.4 Data File Dumper, BBFILE

BBFILE will display the contents of all header blocks and selected data blocks in a blocked binary data file. A short description of what the value represents is shown alongside each standard header block value. A more complete description of the header block values is given in appendix A.

To run BBFILE, type --

```
   $ BBFILE [blocked binary file]
or $ BBFILE <text file>=[blocked binary file],<list>
```

Where:

[blocked binary file] is the name of the blocked binary file to be dumped.

<text file> is the name of the text file in which the blocked binary file's contents will be displayed. The display will appear at the user's terminal if <text file> is not specified.

<list> is an optional list of numbers of the data blocks to be displayed. Dashes may be used to indicate a range of numbers. If <list> is not given on the command line, no data blocks, just the header blocks, will be dumped.

Example:

```
   $ BBFILE MYLIST.TMP =[AGRAM.TESTDATA]EDA.R01,1-4,20
```

10.5 Header Block Changing Program, BWRITE

BWRITE allows a user to alter the contents of the header blocks in a blocked binary data file.

To run BWRITE, type —

```
   $ BWRITE
```

BWRITE, unlike the other AGRAM programs, will prompt the user for instructions.

BWRITE Limitations and Possible Future Inprovements:

Interaction between a user and the BWRITE program is slow and awkward. If a header changing program is going to be used very frequently, other ways of directing the program should be established. One method for providing new or altered header information to a header changing program would be to present an edited BBFILE dump file as input to the header changer (call it CHNGBF).

Example —

```
   $ BBFILE MYLIST.TMP=GOODDATA.R01
   $ EDT MYLIST.TMP      !(EDT is the text-editor used at NSMDC)
       .
       :
   *EXIT
   $ CHNGBF BESTDATA.R01=MYLIST.TMP,GOODDATA.R01
```

Someday we should be able to fill in header information with something like this BWRITE program early in the process. Then any AGRAM program that found its run parameters in the header blocks in its input file would not need to get run parameters from the command line.

10.6 Data File Reformatting program, REFORM

REFORM reformats data files back and forth between blocked binary data files and card-image text files. REFORM will eventually perform a variety of reformatting functions, but at present it can only perform three:

- Reformat BUTTER's card-image output files to blocked binary files that can be input to SCALE
- Reformat PHASE1 card-image output files to blocked binary HIFRIC input files. The PHASE1 files may be in BKY format, the format produced by the PHASE1 program formerly used by the USGS on the BKY operating system at LBL in Berkeley, or they may be in CDMG format, the format produced by the PHASE1 (or counterpart) program in use by the California Bureau of Mines and Geology.
- Reformat SCALE, HIFRIC and/or CORAVD blocked binary files to text files suitable for export to the EDIS archive in Colorado.

To run REFORM, type —

```
$ REFORM [outfile],<msgfile>=[infile],[text file type]
```

Where:

[outfile] is the name of the reformatted output file to be created.

<msgfile> is the name for an optional disk file that will receive the run messages and diagnostics that would normally have gone to an interactive user's terminal or to a batch job's log file.

[infile] is the file that contains the data to be reformatted. There may be several [infile]s when [text file type] = "EDIS".

[text file type] ="BUTTER", "BKY-BUTTER","BKYP1","CDMGP1" or "EDIS" to indicate the type of text file to be read or created.

Example —

```
$ REFORM EDA.BUT=[AGRAM.TESTDATA]ELCDAMIAN.BOU,BUTTER
```

REFORM Limitations and Possible Future Modifications:

REFORM should be capable of reformatting blocked binary data files to a compact text file, complete with summary, for use in disseminating the data to the USGS archive for earthquake related data similarly to the way REFORM does for the EDIS archive.

REFORM should be capable of reformatting EDIS or USGS archive data back into blocked binary files.

REFORM should be capable of reformatting manually digitized data to SCALE input files. Perhaps this function should be performed by a separate program, an IOMTAP/BUTTER counterpart for manually digitized data.

10.7 Reorienting Program, ROTATE

ROTATE reads two input files that represent orthogonal, horizontal components of motion, rotates their orientation, then writes two output files. To run ROTATE, use —

```
    $ ROTATE <out1>,<out2>,<msgfile>=[in1],[in2], -
             [epilat],[epilong]
```

Where <out1> and <out2> are the two output files, <msgfile> is an optional run messages file, [in1] and [in2] are the two input files, and [epilat] and [epilong] are the coordinates of the epicenter. If <out1> and <out2> are not given on the command line, the default output file names will have the same names as the first input file except for the suffixes, which will be .RAD and .TRN.

The resulting orientation of <out1> will be radially away from the epicenter and the orientation of <out2> will be 90 degrees clockwise from <out1>.

Alternatively, the orientations of the output files, [dir1] and [dir2], may be requested explicity on the command line rather than the location of the epicenter. To do so, use —

```
    $ ROTATE [out1],[dir1],[out2],[dir2]=[in1],[in2]
```

ROTATE gets the orientations of the input files, <angle1> and <angle2>, and the recording station co-ordinates, <stnlat> and <stnlong>, from the header blocks in the input files. If this information is missing in the header blocks, it may be given on the command line as in —

```
    $ ROTATE <out1>,<out2>=[in1],<angle1>,[in2],<angle2>, -
             <stnlat>,<stnlong>,[epilat],[epilong]
```

If any of the info is given both on the command line and in the file header, the command line value will be used.

All the directions given to ROTATE should be in degrees clockwise from north. The latitudes should be given as positive numbers for north latitudes, negative numbers for south latitudes; longitudes as positive numbers for east and negative for west.

10.8 BBDATA, a sample program

BBDATA is a sample program that illustrates how to read and write Vinton-Fletcher style blocked binary data files. If you need to write your own program that processes this type of data, you can use a copy of BBDATA as a skeleton to which the new code can be added. BBDATA does nothing but read a blocked binary data file, plot the data, alter the data slightly, then write another blocked binary data file.

BBDATA's subroutine FREAD opens an existing data file and loads all the data into a long array. Subroutine FWRITE creates a new data file and transfers all the data from an array to the file. In many cases, new programs will be able to use FREAD and/or FWRITE without any alteration. FREAD will read AGRAM or DR100 files. When reading a DR100 file, the subroutine will first try to get calibration values from the header blocks in the file; if calibration values are missing from the header blocks, the subroutine will try to get them from a .DBS file if it can find one; if that fails too, it will use default calibration values. If the calibration values are found in the file header, the gain is assummed to be given in decibels, otherwise it is just a factor. FWRITE produces AGRAM style data; that is, the data is on the file in units of cm/sec^2 and is contained in 32-bit real words. DR100 style data is on the file as uncalibrated DR100 counts and is contained in 16-bit integers.

FREAD and FWRITE do the i/o through standard FORTRAN statements. If there is much interest in these subroutines, alternative versions that do the i/o through the more efficient BLKIO subroutines may be added in the future. Instructions for passing arguments to FREAD or FWRITE are given in the first set of comment lines in the subroutines.

The program is stored in the branch VAX in the directory at PUB1:[AGRAM.SAMPLE.BBDATA]. The several files related to the program are:

filename	contents
BBDATA.FOR	FORTRAN code for the main program.
FREAD.FOR	FORTRAN code for the file reading subroutine.
FWRITE.FOR	FORTRAN code for the file writing subroutine.
BBDATA.LNK	Commands for linking the program.
TRY.COM	A command deck that compiles, links and runs the program, then gives the user the opportunity to get a Versatec copy of the plot.
HEADSPACE.INC	An "included" file referenced by the three FORTRAN decks. It allocates array space and defines constants used while manipulating header blocks.
BBDATA.ALL	A copy of all the above files gathered together into a single file for convenient printing.

Object code for the FREAD and FWRITE subroutines is also available from the AGRAM general subroutine library at PUB1:[AGRAM.SMASHLIB]SMASHLIB.OLB.

APPENDIX A

Contents of Data File Header Blocks

A blocked binary data file generated by the AGRAM programs contains at least two blocks of auxiliary data at the beginning of the file. The first header block contains integer items, the second contains real items. Most of the locations in the first two header blocks are reserved for specific information as shown in the table below. The locations were assigned according to an organization originally established for DR100 data files with requirements for AGRAM files fitted in later. Many of the locations reserved for DR100 files are not used at all by AGRAM programs and many new locations were assigned in the "scratch" area to accomodate AGRAM data. The table identifies all reserved locations. Those locations that are actually set or read by any of the AGRAM programs are shown with a leading asterisk.

<u>First Integer Header Block, 256 (2-byte) Elements Long</u>

*(1)	Number of optional integer header blocks that follow this one.
*(2)	Number of optional ASCII header blocks that follow the real header blocks.
*(3)	Integer value that represents "undefined" (usually = -32768).
*(4)	Real or integer data flag; = 1 if each data block contains 128 real (4-byte) elements, = item (3) if each data block contains 256 integer (2-byte) elements.
(10)	Year of the event
(11)	Julian day
(12)	Hour
(13)	Minute
(14)	Second
(15)	Millisecond
(16)	microsecond.
(17)	Sample number of first time mark
(20)	Serial number
(21)	Event number
(30)	Number of components recorded with (and including) this one.
*(31)	Number of data blocks (without headers).
*(32)	Index of the last sample in the last data block.

(40)	'FB' or 'VT' (force-balance accelerometer or velocity transducer)
*(41)	Vertical orientation, angle in degrees measured from vertical. This item will be 90 in data files representing horizontal motion. Note that before November 1982, this element was set to 'HR' or 'VR' to indicate whether the time series data represents horizontal or vertical motion.
*(42)	Horizontal orientation, a value between 0 and 360 degrees clockwise from north. This item will be 0 in data files representing vertical motion.
*(43)	'AC', 'VL', or 'DP' (acc, vel, or disp) or 'BU' for BUTTER data, 'IR' for SCALE data.
(44)	This item is no longer used. Before November 1982, it was set to + or - 1 to indicate polarity of vertical motion. Vertical motion is now described by item (41).
*(200-256)	Scratch area.

Scratch locations used in (REFORMatted) BUTTER files:

*(200)	Number of digitized traces in the file. (There are usually 3 data traces, 1 or 2 reference traces, and 1 time tick trace.)
*(201)	Type of trace, 1-> time-tick, 2-> reference line, and 3-> data.
*(202)	Number of the first data block for the trace.
*(203)	Number of data blocks for the trace.
*(204-?)	Repeat items (201),(202), and (203) for each trace.

Second Header Block, 128 Real (4-byte) Elements Long

*(1)	Number of optional real header blocks that follow this one.
*(2)	Real value that represents "undefined" (usually =-0.3e-38).
*(5)	Sampling rate, samples/second. If undefined, then it's output data from BUTTER or SCALE.
(10)	Earthquake or explosion latitude, degrees.
(11)	Earthquake or explosion latitude, minutes.
(12)	Earthquake or explosion longitude, degrees.
(13)	Earthquake or explosion longitude, minutes.
(14)	Depth in kilometers
(15)	Magnitude
(16)	Moment
(17)	Weight of explosives
(18)	Origin time
(19)	Distance from station to epicenter
(20)	Azimuth of station from epicenter
(21)	Takeoff angle

(40)	Instrument latitude, degrees
(41)	Instrument latitude, minutes
(42)	Instrument longitude, degrees
(43)	Instrument longitude, minutes
(44)	Instrument altitude
(45)	Voltage input
*(46)	Digitizer output, digitization units/volt for DR100 data, digitization units/cm for SMA data.
*(47)	Anti-alias filter's corner frequency, cps. (in AGRAM files, this is the freqency used with the combined instrument correction and anti-alias filter in HIFRIC. Other high-cut filters may be applied in CORAVD, but those filter characteristics are described in items (105) through (113).
*(48)	Anti-alias filter's rolloff bandwidth, cps.
*(49)	Natural period of an SMA transducer, in seconds; or natural frequency of a DR100 transducer.
*(50)	Damping coefficient of the transducer, fraction of critical damping.
*(51)	Coil constant (volts/cm/sec/sec) for DR100 data, or Recorder sensitivity (cm/g) for SMA data.
(52)	Gain of a DR100 amplifier (volts/volt).
*(60)	Clock correction, seconds
(70-71)	P and S wave picks
*(90-128)	Scratch area

Scratch locations filled in SCALE, HIFRIC, and CORAVD output files —

*(90)	Time of the first data point, not including clock correction (item 60), in seconds. (usually =0.0)
*(91)	Value of the first data point, in cm/sec/sec.
*(92)	Time of the last data point.
*(93)	Value "
*(94)	Time of the maximum value.
*(95)	Value "
*(96)	Time of the minimum value.
*(97)	Value "
*(98)	Offset of SCALE data. This value is subtracted from all data items in HIFRIC.
*(103)	Source-of-data indicator: 1.0 if from SCALE; 2.0 if from HIFRIC, with interpolation only; 2.1 if from HIFRIC, with time domain instrument correction and high-cut filter; 2.2 if from HIFRIC, with frequency domain instrument correction and high-cut filter; 3.0 if from CORAVD, without low-cut filter or linear baseline correction to velocity; 3.1 if from CORAVD, without filter, with linear baseline correction;

3.2 if from CORAVD with filter and linear baseline correction.

*(99)	Beginning time for the least squares fit baseline correction.
*(100)	Ending time for the least squares fit baseline correction.
*(101)	Slope of the linear baseline correction.
*(102)	Intercept of the linear baseline correction.
*(104)	Low cut filter cut off, cps.
*(105)	High-cut filter cut off, cps.
*(106)	Low-cut filter rolloff band width,
*(107)	High-cut filter rolloff band width, cps or integer rolloff order/2.
*(108)	Low-cut filter type,
*(109)	High-cut filter type, =1.,2.,3. or 4. to indicate unidirectional Butterworth filter, bidirectional Butterworth filter, Ormsby filter, or FFT filter.
*(110)	Low-cut filter padding identifier,
*(111)	High-cut filter padding indentifer, (1.=> zero pad, 2.=> cosine taper, ... others, too)
*(112)	Low-cut Ormsby filter option,
*(113)	Reserved for the high-cut Ormsby filter option, although such is not included in the AGRAM package yet.

APPENDIX B

Examples

Two sets of sample AGRAM commands are listed in this appendix. The first set shows the commands used for preliminary routine processing of a 3-component record. The second set shows more detailed processing of a single component than was done in the first set. Both sets process the same record, that from the Pleasant Valley pumping plant basement during the mainshock of the Coalinga earthquake on 02 May 83. Run messages and plots resulting from the second set of commands are also shown in this appendix.

B.1 First Example

The commands that are printed out below are used for the first pass in the routine processing of incoming data. The items that are circled on the print-out are those that need to be changed according to the data being processed. Note that no filtering is requested in the CORAVD commands shown here. Using the plotted results from these CORAVD and FASPLT commands, the user must decide whether or not the CORAVD and FASPLT steps should be rerun with filtering applied in CORAVD. During the routine processing of the data used in this example, CORAVD was rerun with a filter as is shown in the second example. The response spectra program is not run during the first pass either, since it is a time consuming process and the decision whether to filter or not should be made beforehand.

These commands are arranged to be run in "indirect command mode" on the VAX, with the commands coming from a preassembled file and the results going to an interactive user's terminal. The commands in the indirect command file are constructed just like the commands a user would type directly in interactive mode except that the dollar sign the VAX would have given as a prompt to an interactive user is typed by the user rather than by the machine.

Before running these commands, the user must physically mount the IOM/TOWILL tape on the tape handler labeled MTA1:.

Here follow the commands.

```
$on warning then goto end
$ DISM MTA1:/NOUNL
$ MOUNT MTA1:/FOR/BLOCKSIZE=4000
$ IOMTAP TEMP.IOM=MTA1:1-6
$ BUTTER
 NONE                        !read run parameters from this deck
 NONE                        !want run messages to terminal, not disk
 PUB1:[AGRAM.TESTDATA]SAMPLE.IOM   != the input data file
 NONE                        !  no need for calibration file
 TEMP.BOU                    != name of the output data file
 YES                         !yes, want bbf output file
! a nothing-but-comment line here, just for illustration
  -400                       != min forward step size
 YES                         !yes, butting lines removed
 NO                          !no hanning smoothing
 NO                          !no decimation
 NO                          !no sampling
 YES                         !yes, options are correct
 SDDDRRTB                    !traces in the first frame
 DDDRRTBB                    !              second
 DDDRRTBB                    !                :
 DDDRRTBB                    !
 DDDRRTBB                    !
 DDDRRTB                     !              last
 DONE                        !end of trace descriptions
 YES                         !yes, traces are correct
$ REFORM TEMP.BUT=TEMP.BOU,BUTTER
$ SCALE TEMP.=TEMP.BUT,0.5,1.85,1.93,1.81
$ STSP TEMP.R01,.R02,.R03
$ HIFRIC TEMP.G01=TEMP.R01,.039,.60
$ HIFRIC TEMP.G02=TEMP.R02,.040,.60
$ HIFRIC TEMP.G03=TEMP.R03,.039,.60
$ STSP TEMP.G01,.G02,.G03
$ CORAVD TEMP.=TEMP.G01
$ CORAVD TEMP.=TEMP.G02
$ CORAVD TEMP.=TEMP.G03
$ STSP TEMP.A01,TEMP.V01,TEMP.D01
$ STSP TEMP.A02,TEMP.V02,TEMP.D02
$ STSP TEMP.A03,TEMP.V03,TEMP.D03
$ SFAS TEMP.A01
$ SFAS TEMP.A02
$ SFAS TEMP.A03
$end:
```

B.2 Second Example

The second example printed below illustrates more optional aspects of the programs than did the first sample. Comments in the print-out indicate the purpose of the commands that follow the comments. Comments begin with an exclamation mark (!). Those lines that contain nothing but comment contain a dollar sign followed by an exclamation mark ($!) in the first two columns.

These commands are arranged to be run in batch mode on the VAX machine. They are constructed similarly to those in the first sample, although a few commands used to control the execution of the batch job have been added. Those commands that are only used for batch job control are shown in lower case and those commands an interactive user might also use are in upper case. For this example, the IOMTAP function has been performed separately so the rest of the job may run at night when there is no one around to hang a tape.

These commands show VTSP, the Versatec version of TSPLOT, as the plotting program. VTSP is a good plotting program for batch mode use because it does not require any user interaction, but plots from the Versatec equipment at NSMDC are not as good quality as those from the Calcomp. Better quality plots could be drawn with the Calcomp plotter using CTSP or ITSP and the device-independent VIEWER software. ITSP, VIEWER, and the NSMDC Calcomp equipment require user interaction, however, and they are unique to the NSMDC site, so they aren't used as the plotting method in this appendix.

Here are the commands.

```
                           :
$set default publ:[converse.test]
$define sys$print dummy
$@[agram]agram.def
$set verify
$on warning then goto end
$!
$!     This is [AGRAM.TESTCASES]APPB.BAT.
$!     It illustrates the AGRAM programs by processing one of the
$!       horizontal components from the first four frames of the
$!       sample data (Coalinga EQ, Pleasant Valley pumping plant).
$!       It is quite a long record, almost 60 seconds in 6 frames,
$!       but only the first 4 frames are processed here.
$!     Submit this deck to run in the evening with:
$!       $ SUBMIT/AFTER=18:00  APPB.BAT
$!
$!
$!   The IOMTAP with its DISM and MOUNT commands have been
$!   commented out ("$! +++") here.  These commands were executed
$!   separately so we won't have to mount a tape to run this
$!   example.  The BUTTER below uses the results from IOMTAP that
$!   have already been saved in the file at
$!   PUB1:[AGRAM.TESTDATA]SAMPLE.IOM.
$!
$!
$! +++ $ DISM MTA1:/NOUNL
$! +++ $ MOUNT MTA1:/FOR/BLOCKSIZE=4000
$! +++ $ IOMTAP SAMPLE.IOM=MTA1:,1-6
$!
$!
$!   Run BUTTER.  Although there are 6 frames of data on the .IOM
$!   file, let this BUTTER run process the first 4 frames only.
$!
$!
$ BUTTER
 NONE                      !read run parameters from this deck
 NONE                      !want run messages to terminal, not disk
 PUB1:[AGRAM.TESTDATA]SAMPLE.IOM   != the input data file
 NONE                              !  no need for calibration file
 PVB.BOU                           != name of the output data file
 YES                               !yes, want bbf output file
! a nothing-but-comment line here, just for illustration
  -400                             != min forward step size
 YES                               !yes, butting lines removed
 NO                                !no hanning smoothing
 NO                                !no decimation
 NO                                !no sampling
 YES                               !yes, options are correct
 SDDDRRTB                          !traces in the first frame
 DDDRRTBB                          !               second
 DDDRRTBB                          !               third
 DDDRRTB                           !               fourth
 DONE                              !end of trace descriptions
 YES                               !yes, traces are correct
$!
$!
$!   Reformat butter output from a text file to a blocked binary
$!   file.  (This step won't be necessary once the BUTTER program
$!   is updated.)
$!
$!
$ REFORM PVB.BUT=PVB.BOU,BUTTER
$!
$!
$!   Run SCALE to get scaled, uncorrected data in three files
$!   (pvb.r01, pvb.r02, and pvb.r03) corresponding to the
$!   three accelerometer traces on the original record.
$!
$!   Note that the first time coordinate in each trace is given
$!   the value of zero even though the three traces do not have
$!   precisely the same horizontal coordinate in their first
$!   digitized sample.  The arrivals of each trace on the
$!   original recording film, simultaneous with the lighting of
$!   bulb filament, ARE synchronous.  Their physical alignment
$!   across the width of the film need not be identical.
$!
$!   Plot results for the third trace, the horizontal component
$!   oriented at N45E.
$!
$!
$ SCALE PVB.=PVB.BUT,0.5,1.85,1.93,1.81
$ VTSP PVB.R03,,,20.0,0.3
$!
$!
$!   Run HIFRIC for the third trace to get interpolated,
$!   high-cut filtered and instrument-corrected data.
$!   Plot results.
$!
$ HIFIRC PVB.G03=PVB.R03,.039,.60
$ VTSP PVB.G03,,,,0.3
$!
$!
$!   Compare the effects of the interpolation, high-cut
$!   filter and instrument correction applied in HIFRIC to
$!   uninterpolated, uncorrected data produced by SCALE.
$!   Also compare these to interpolated but uncorrected data.
$!   To do so, rerun HIFRIC with interpolation only, no
$!   instrument correction or filter, then plot two seconds of
$!     (1) uncorrected (SCALE) data,
$!     (2) interplolated (HIFRIC with INTERP) data, and
$!     (3) high-cut filtered, instrument corrected (standard
$!         HIFRIC) data
$!   with an exaggerated scale to illustrate how the  peak value
$!   was affected by processing.
$!
$!
$ HIFRIC PVB.I03=PVB.R03,INTERP
$ VTSP PVB.R03,.I03,.G03, 6.0,8.0,2.0
$!
$!
$!   The third curve in the plot above shows that the high frequency
$!   digitization noise in the original data near the peak located at
$!   about 7.2 seconds has been amplified by the instrument correction.
$!
$!   The BBFILE program can be used to examine the contents of
$!   the data files near the peak at 7.2 seconds.  The data at 7.2
$!   seconds is in block 12 of the HIFRIC output file and, if
$!   approximately 600 samples per centimeter were digitized, the
$!   data at 7.2 seconds would be in or near block 68 of the SCALE
$!   output file, and in this case it is in block 69.
$!   (block 12 is determined from 11.2 = 7.2 seconds * 200 samples
$!    per second / 128 samples per block.)
$!   (block 68 is determined from 67.5 = 7.2 seconds * 600
$!    samples per second * 2 items (x and y) per sample
$!    / 128 items per block.)
$!
$!
$ BBFILE PVB.G03,12
$ BBFILE PVB.R03,68-69
$!
$!
$!   Notice that the frequency-domain instrument correcting
$!   algorithm has nearly the same effect as the time-domain
$!   algorithm.
$!
$!
$ HIFRIC PVB2.G03=PVB.R03,.039,.60,FDIC
$ VTSP PVB.G03,PVB2.G03,  6.0,8.0,2.0
$!
$!
$!   Run CORAVD with linear baseline correction to the velocity
$!   and without long-period filtering.  Plot results.
$!
$!
$ CORAVD PVB.=PVB.G03
$ VTSP PVB.A03,.V03,.D03
$!
$!
$!   Run CORAVD with long-period filtering but no linear
$!   baseline correction and plot results.
$!
$!
$ CORAVD PVB2.=PVB.G03,0.0,0.0,BI,0.100,2
$ VTSP PVB2.A03,.V03,.D03
$!
$!   Run FASPLT to calculate and plot Fourier amplitude
$!   spectrum.
$!
$ VFAS PVB2.A03,ZCROSS,NONOISE,LOGLOG,LINLIN
$!
$!
$!   Run PHASE3 to calculate relative response spectra, then
$!   run PH3PLT to plot results.
$!
$!
$ COPY PVB2.A03 PVB2.A01  ! phase3 expects to process 3 files
$ COPY PVB2.A03 PVB2.A02  !   "
$ COPY PUB1:[AGRAM.TESTCASES]NOTITLE.TXT PVB2.TIT
$ PHASE3
PVB2
PVB2.PH3
$ PH3PLT
PVB2.PH3
2
1
$!
$!   Now run VIEWERR and RASM to get a hard-copy plot from the
$!   batch.plt file generated by PH3PLT.
$!
$ RUN PUB1:[BGMF]VIEWERR
PLOT BATCH.PLT
EXIT
```

```
$ MCR RASM
$ DELETE *.PLV;*
$!
$!
$!   How many data files were generated?  They should either
$!   be archived or deleted.
$!
$!
$ dir PVR.*,PVR2.*
$!
$!
$!   end of job.
$!
$end: continue
$eoj
```

B.3 Results from the Second Example

Here follow most of the run messages and all of the plots generated by the command deck printed above, in section B.2.

```
$on warning then goto end
$!
$!      This is [AGRAM.TESTCASES]APPB.BAT.
$!      It illustrates the AGRAM programs by processing one of the
$!        horizontal components from the first four frames of the
$!        sample data (Coalinga EQ, Pleasant Valley pumping plant).
$!        It is quite a long record, almost 60 seconds in 6 frames,
$!        but only the first 4 frames are processed here.
$!      Submit this deck to run in the evening with:
$!        $ SUBMIT/AFTER=18:00  APPB.BAT
$!
$!
$!    The IOMTAP with its DISM and MOUNT commands have been
$!    commented out ("$! +++") here.  These commands were executed
$!    separately so we won't have to mount a tape to run this
$!    example.  The BUTTER below uses the results from IOMTAP that
$!    have already been saved in the file at
$!    PUB1:[AGRAM.TESTDATA]SAMPLE.IOM.
$!
$!
$! +++ $ DISM MTA1:/NOUNL
$! +++ $ MOUNT MTA1:/FOR/BLOCKSIZE=4000
$! +++ $ IOMTAP SAMPLE.IOM=MTA1:,1-6
$!
$!
$!    Run BUTTER.  Although there are 6 frames of data on the .IOM
$!    file, let this BUTTER run process the first 4 frames only.
$!
$!
$ BUTTER

     BUTTER will read its run parameters from the batch input stream.
     It will issue prompts as though it were asking an interactive
     user for the parameters.  If all the prompts are answered with
     a blank line, a sample BUTTER run will result.

 .. Enter the name of an optional input disk file containing your
    answers to all subsequent prompts from BUTTER.
    (<CR> or "none" => continue to prompt the terminal) ..
 NONE                        !READ RUN PARAMETERS FROM THIS DECK
 .. Enter the name of an optional output disk file to contain run
    messages.  (<CR> or "none" => display run messages at the
    terminal) ..
 NONE                        !WANT RUN MESSAGES TO TERMINAL, NOT DISK
 .. Enter the name of the input disk file containing data read from
    an IOM/TOWILL tape.  (<CR> = PUB1:[agram.testdata]sample.iom) ..
PUB1:[AGRAM.TESTDATA]SAMPLE.IOM    != THE INPUT DATA FILE
```

```
 .. Enter the name of an optional calibration data file.
    (<CR> or "none"=> the data doesn t require new calibration
    values) ..
 NONE                                 !  NO NEED FOR CALIBRATION FILE
 .. Enter a name for the output data file.  (<CR> = temp.but) ..
 PVB.BOU                              != NAME OF THE OUTPUT DATA FILE

    The output file can be a blocked binary file that can be read  by
    the SCALE program, or the file can be a text file that can be
    read and edited with a text editor.  A text output file must be
    reformatted to a blocked binary file by the REFORM program
    before it can be read by the SCALE program.
 .. Shall the output file be a blocked binary file?  (<CR> = "yes") ..
 YES                                  !YES, WANT BBF OUTPUT FILE
 .. Enter minimum forward step size in microns.  Give a negative
    value to allow backstepping (<CR>=-400) ..
! A NOTHING-BUT-COMMENT LINE HERE, JUST FOR ILLUSTRATION
  -400                                != MIN FORWARD STEP SIZE
 .. Do you want points beyond the butting lines removed
    (<CR> = "yes") ..
 YES                                  !YES, BUTTING LINES REMOVED
 .. Do you want Hanning smoothing? (<CR> = "no") ..
 NO                                   !NO HANNING SMOOTHING
 .. Do you want to decimate close points? (<CR> = "no") ..
 NO                                   !NO DECIMATION
 .. Do you want to sample the data as we go?
    (<CR> = "no") ..
 NO                                   !NO SAMPLING


Options in effect:


      Minimum step =  -400 microns
      Removal of points beyond butting lines
    NO Hanning smoothing
    NO Decimation of close points
    NO Data sampling

 .. Are these all correct?  (<CR> = "yes") ..
 YES                                  !YES, OPTIONS ARE CORRECT
```

Now BUTTER needs information about the various traces in the separately digitized frames. You must identify each trace as type D, R, T, B or S. D for a data trace, R for a reference trace, T for a time-tick trace, B for a butting line, and S for a trace to be skipped. You will be asked to enter the trace types as a single string of characters for each frame. Respond by typing the characters, without any spaces, in the order the traces are given on the IOM tape. (Example: SDDDRRT) This trace order information is given in the cover sheet IOM/TOWILL provides along with the tape. After you have given the trace types for all the frames, respond with DONE to the next prompt for trace types.

```
.. Enter trace types for frame  1 (<CR>= SDDDRRT) ..
SDDDRRTB                          !TRACES IN THE FIRST FRAME
.. Enter trace types for frame  2 (<CR>= DONE) ..
DDDRRTBB                          !              SECOND
.. Enter trace types for frame  3 (<CR>= DONE) ..
DDDRRTBB                          !              THIRD
.. Enter trace types for frame  4 (<CR>= DONE) ..
DDDRRTB                           !              FOURTH
.. Enter trace types for frame  5 (<CR>= DONE) ..
DONE                              !END OF TRACE DESCRIPTIONS
   4Traces.
      Trace types for frame  1 = SDDDRRTB
                             2 = DDDRRTBB
                             3 = DDDRRTBB
                             4 = DDDRRTB
.. Are these all correct? (<CR> = "yes") ..
YES                               !YES, TRACES ARE CORRECT
```

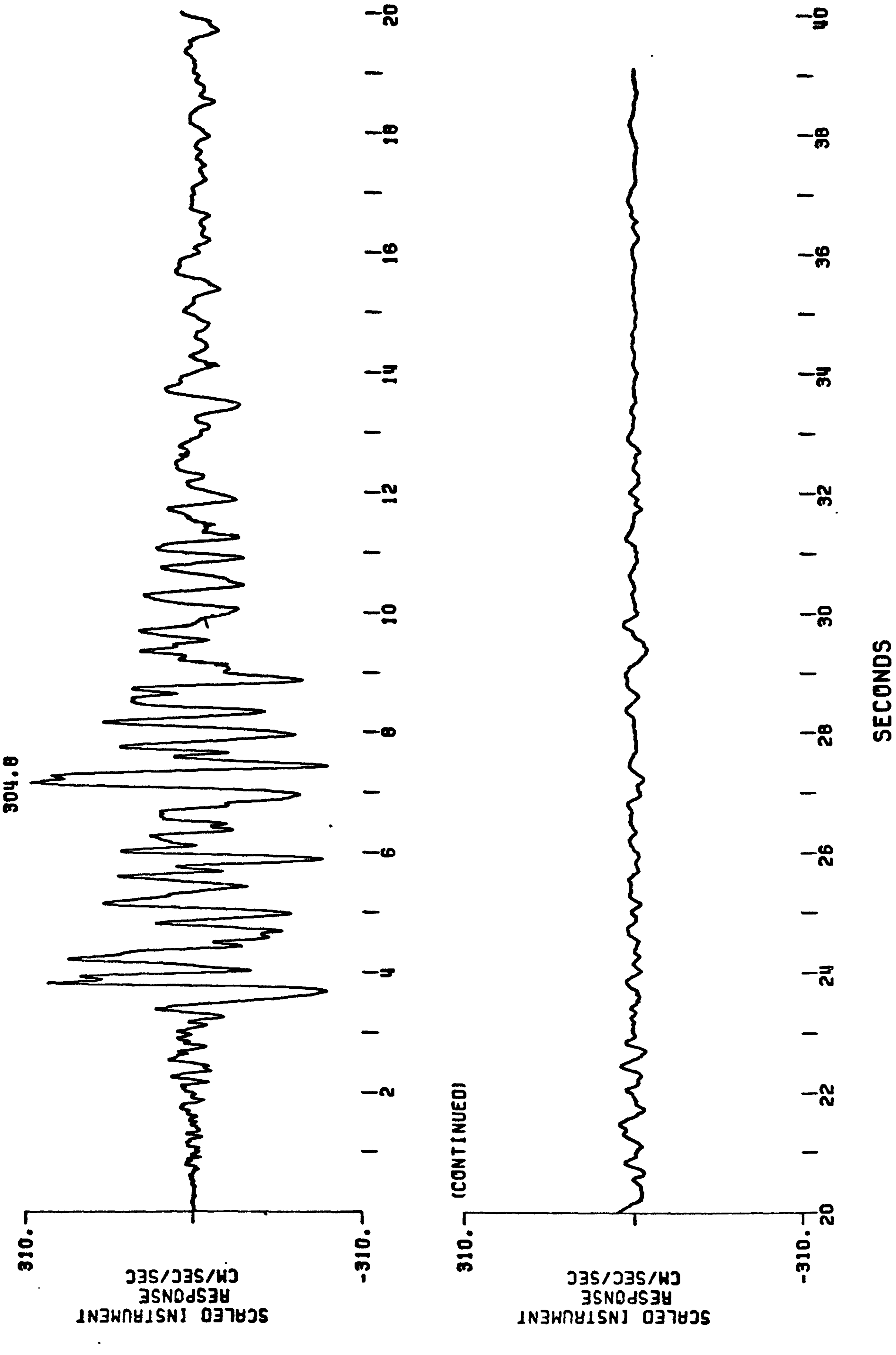
BUTTER now has all the run parameters it needs. It will not issue any more prompts during this run.

```
   Frame  1, trace  1, a trace to skip:
          2 points read from the input file.
      Trace skipped.
   Frame  1, trace  2, a data trace:
       6620 points read from the input file.
      There are    6 backstepping points included in the trace, the largest
      of which backstepped      5 digitzing units.
   Frame  1, trace  3, a data trace:
       6560 points read from the input file.
      There are   19 backstepping points included in the trace, the largest
      of which backstepped      3 digitzing units.
   Frame  1, trace  4, a data trace:
       6405 points read from the input file.
      There are    7 backstepping points included in the trace, the largest
      of which backstepped      4 digitzing units.
   Frame  1, trace  5, a reference trace:
       6789 points read from the input file.
   Frame  1, trace  6, a reference trace:
       6625 points read from the input file.
   Frame  1, trace  7, a time-tick trace:
         19 points read from the input file.
   Frame  1, trace  8, a second butting line:
        800 points read from the input file.
      There are  338 backstepping points included in the trace, the largest
      of which backstepped     22 digitzing units.

   This is the first frame of a multi-frame record
      Initial orientation:
         Weighted average slope of the  2 reference trace(s) =  1.3311380E-0

      Initial orientation to the left of the RH butting line:
         Weighted average slope of the  2 reference trace(s) =  1.5903132E-C
      Minimum X subtracted from the traces =   -52900
   Reference trace intersects the RH butting line
      at (  93438.51     ,  3415.444    ).
   Final orientation after butting:
      Weighted average slope of the  2 reference trace(s) =  8.3469074E-05

   Data      trace  1 contains  5809 points.
   Data      trace  2 contains  5766 points.
   Data      trace  3 contains  5735 points.

Frame  2, trace  1, a data trace:
    7602 points read from the input file.
Frame  2, trace  2, a data trace:
    7565 points read from the input file.
   There are    1 backstepping points included in the trace, the largest
   of which backstepped      4 digitzing units.
Frame  2, trace  3, a data trace:
    7387 points read from the input file.
Frame  2, trace  4, a reference trace:
    7584 points read from the input file.
Frame  2, trace  5, a reference trace:
    7578 points read from the input file.
Frame  2, trace  6, a time-tick trace:
      19 points read from the input file.
Frame  2, trace  7, a first butting line:
     786 points read from the input file.
   There are  357 backstepping points included in the trace, the largest
   of which backstepped     28 digitzing units.
Frame  2, trace  8, a second butting line:
     815 points read from the input file.
   There are  357 backstepping points included in the trace, the largest
   of which backstepped     16 digitzing units.

This is an intermediate frame of a multi-frame record
   Initial orientation:
      Weighted average slope of the  2 reference trace(s) =  1.5793098E-03
   Initial orientation to the right of the LH butting line:
      Weighted average slope of the  2 reference trace(s) =  1.5161113E-03
   Butting line differences before rotation:
      The slope   difference =  2.1899128E-03 from frame 1 to frame 2.
      The angular difference =  2.1899093E-03 from frame 1 to frame 2.
   Butting line differences after rotation:
      The slope   difference = -1.7568306E-07 from frame 1 to frame 2.
      The angular difference = -1.7568306E-07 from frame 1 to frame 2.
   Final orientation to the right of the LH butting line:
      Weighted average slope of the  2 reference trace(s) =  4.8249527E-04
   Final orientation after butting:
      Weighted average slope of the  2 reference trace(s) =  6.9275961E-04

   Data      trace  1 contains  6004 points.
   Data      trace  2 contains  6027 points.
   Data      trace  3 contains  5906 points.

Frame  3, trace  1, a data trace:
    7711 points read from the input file.
Frame  3, trace  2, a data trace:
    7640 points read from the input file.
Frame  3, trace  3, a data trace:
    7579 points read from the input file.
Frame  3, trace  4, a reference trace:
    7665 points read from the input file.
Frame  3, trace  5, a reference trace:
    7557 points read from the input file.
Frame  3, trace  6, a time-tick trace:
      19 points read from the input file.
Frame  3, trace  7, a first butting line:
     824 points read from the input file.
   There are  374 backstepping points included in the trace, the largest
   of which backstepped     19 digitzing units.
Frame  3, trace  8, a second butting line:
     809 points read from the input file.
   There are  365 backstepping points included in the trace, the largest
   of which backstepped     24 digitzing units.

This is an intermediate frame of a multi-frame record
   Initial orientation:
      Weighted average slope of the  2 reference trace(s) =  4.5525673E-03
   Initial orientation to the right of the LH butting line:
      Weighted average slope of the  2 reference trace(s) =  4.6221987E-03
   Butting line differences before rotation:
      The slope   difference =  4.3730285E-03 from frame 2 to frame 3.
      The angular difference =  4.3730006E-03 from frame 2 to frame 3.
   Butting line differences after rotation:
      The slope   difference =  3.6955825E-10 from frame 2 to frame 3.
      The angular difference =  3.6955825E-10 from frame 2 to frame 3.
   Final orientation to the right of the LH butting line:
      Weighted average slope of the  2 reference trace(s) =  2.5460196E-03
   Final orientation after butting:
      Weighted average slope of the  2 reference trace(s) =  2.6780460E-03

   Data      trace  1 contains  6026 points.
   Data      trace  2 contains  6026 points.
   Data      trace  3 contains  5988 points.
   Reference trace  1 contains  6010 points.

Frame  4, trace  1, a data trace:
    7683 points read from the input file.
Frame  4, trace  2, a data trace:
    7658 points read from the input file.
Frame  4, trace  3, a data trace:
    7639 points read from the input file.
Frame  4, trace  4, a reference trace:
    7666 points read from the input file.
Frame  4, trace  5, a reference trace:
    7622 points read from the input file.
Frame  4, trace  6, a time-tick trace:
      19 points read from the input file.
Frame  4, trace  7, a first butting line:
       793 points read from the input file.
     There are  351 backstepping points included in the trace, the largest
     of which backstepped     28 digitzing units.

  This is the final frame of a multi-frame record
     Initial orientation:
        Weighted average slope of the  2 reference trace(s) =  2.2990631E-03
     Initial orientation to the right of the LH butting line:
        Weighted average slope of the  2 reference trace(s) =  2.2672640E-03
     Butting line differences before rotation:
        The slope   difference =  2.9948236E-03 from frame 3 to frame 4.
        The angular difference =  2.9948146E-03 from frame 3 to frame 4.
     Butting line differences after rotation:
        The slope   difference =  4.0259327E-08 from frame 3 to frame 4.
        The angular difference =  4.0259327E-08 from frame 3 to frame 4.
     Final orientation after butting:
        Weighted average slope of the  2 reference trace(s) =  8.5137029E-04

     Data      trace  1 contains  6850 points.
     Data      trace  2 contains  6822 points.
     Data      trace  3 contains  6817 points.
     Reference trace  1 contains  6835 points.

*** Sorry, BUTTER cannot generate blocked-binary data files
    yet.  Output will be in a text file that can be reformatted
    with the REFORM program to get a blocked-binary data file. ***

  Writing butted data in text form to the output file named
     PVB.BOU
     The time trace has   76 points.
     Trace  1 has 24689 data points and 24791 reference points.
     Trace  2 has 24641 data points and 24791 reference points.
     Trace  3 has 24446 data points and 24791 reference points.
```

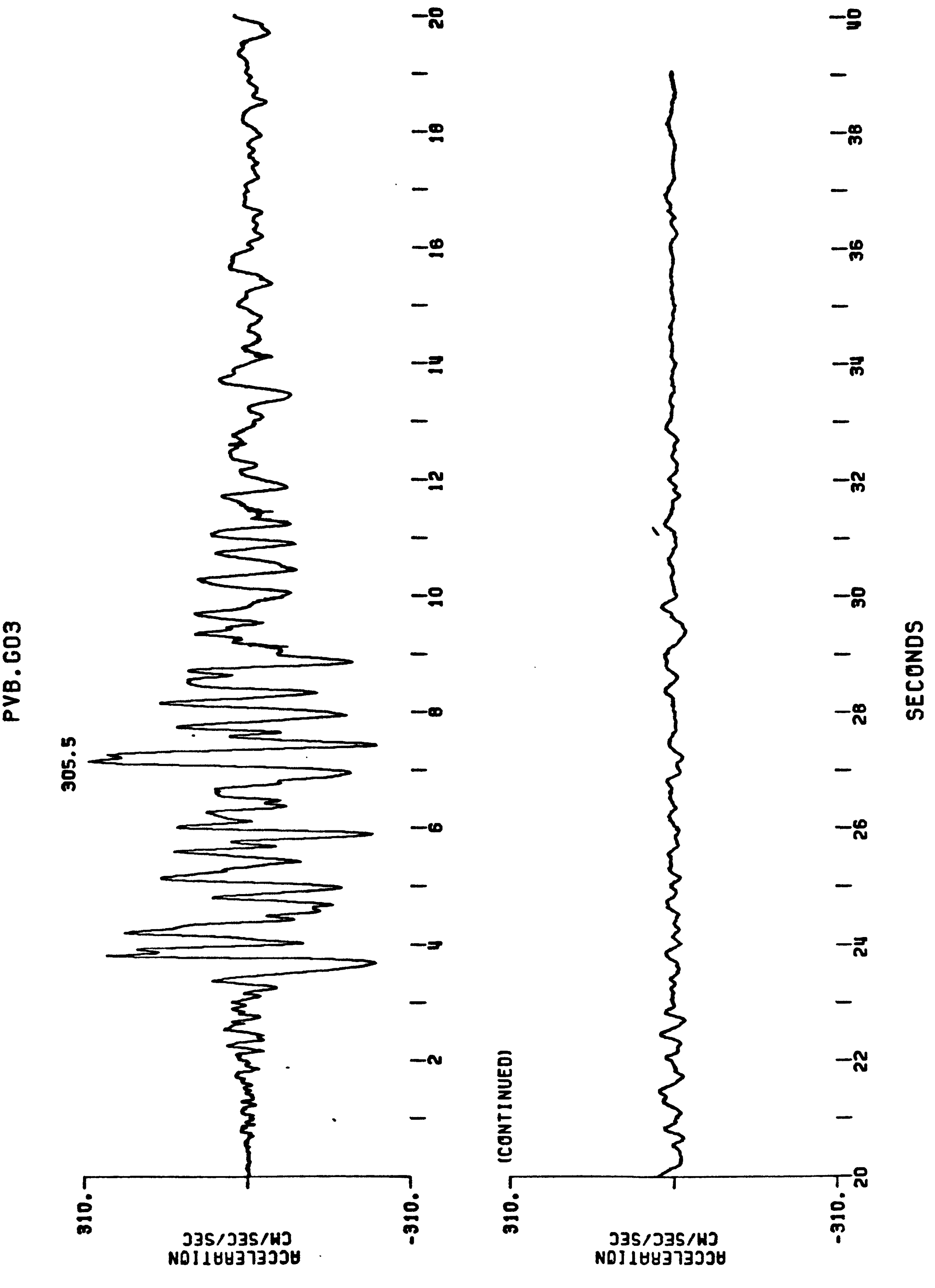
```
$!
$!
$!   Reformat butter output from a text file to a blocked binary
$!   file.  (This step won't be necessary once the BUTTER program
$!   is updated.)
$!
$!
$ REFORM PVB.BUT=PVB.BOU,BUTTER
    REFORM, 31aug83 version.
    The first 3 lines of the butter file are --
    Final BUTTER data traces
     no title                                        Time trace

    Number of time marks=    76
    Component 1
      24689 samples in the data trace, and
      24791 samples in the reference trace.
    Component 2
      24641 samples in the data trace, and
      24791 samples in the reference trace.
    Component 3
      24446 samples in the data trace, and
      24791 samples in the reference trace.
```

```
$!
$!
$!   Run SCALE to get scaled, uncorrected data in three files
$!   (pvb.r01, pvb.r02, and pvb.r03) corresponding to the
$!   three accelerometer traces on the original record.
$!
$!   Note that the first time coordinate in each trace is given
$!   the value of zero even though the three traces do not have
$!   precisely the same horizontal coordinate in their first
$!   digitized sample.  The arrivals of each trace on the
$!   original recording film, simultaneous with the lighting of
$!   bulb filament, ARE synchronous.  Their physical alignment
$!   across the width of the film need not be identical.
$!
$!   Plot results for the third trace, the horizontal component
$!   oriented at N45E.
$!
$!
$ SCALE PVB.=PVB.BUT,0.5,1.85,1.93,1.81

    SCALE, 23apr84 version.

    Input file = PVB.BUT
       digitization scale = 10000. units/cm.
       time-tick interval = 0.50 seconds.

    Output file = PVB.R01
       recorder sensitivity = 1.8500 cm/g.
       There are 24689 points in the trace.
       Trace length = 39.126 seconds.
       Mean value =  0.21769E+01 cm/sec**2.
       Max. value =  0.27011E+03 at  3.743 seconds.
       Min. value = -0.17763E+03 at  5.270 seconds.
       There are  15 back-stepping points, the largest of which is for
          0.00051 seconds at time =  4.474 seconds.
       The largest difference between two consecutive, unsmoothed
          time-tick intervals was  -2.981% at tick number 49.
       There was a  -1.230% difference betwen the first ( 0.4961 cm.)
          and last ( 0.5022 cm) unsmoothed tick intervals.
       The first data point (at 0.000 sec.) occurred before the first
          time tick (at  0.021 sec.).
       The last data point (at 39.126 sec.) occurred after the last
          time tick (at 37.021 sec.).
       The last point in the reference trace (at 39.091 sec.) occurred
          before the last point in the data trace (at 39.126 sec.), so
          the reference line was extended with the slope
          (=-0.25944E-02) of the last centimeter of  reference trace.

    Output file = PVB.R02
       recorder sensitivity = 1.9300 cm/g.
       There are 24641 points in the trace.
       Trace length = 39.061 seconds.
       Mean value =  0.45270E+01 cm/sec**2.
       Max. value =  0.17053E+03 at  4.761 seconds.
```

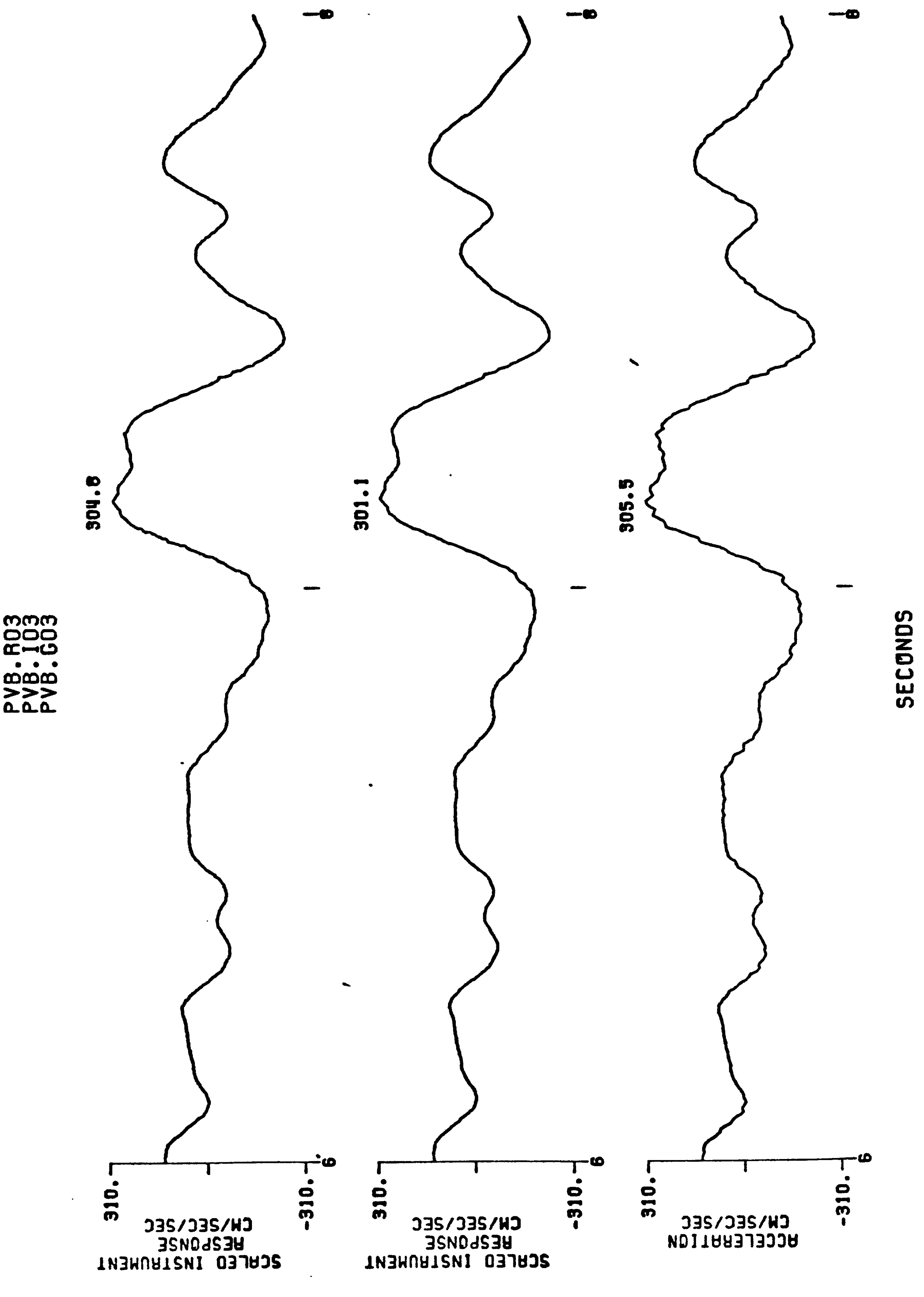
```
          Min. value = -0.20974E+03 at  3.984 seconds.
          There are  58 back-stepping points, the largest of which is for
             0.00030 seconds at time =  4.156 seconds.
          The largest difference between two consecutive, unsmoothed
             time-tick intervals was  -2.981% at tick number 49.
          There was a  -1.230% difference betwen the first ( 0.4961 cm.)
             and last ( 0.5022 cm) unsmoothed tick intervals.
          The first data point (at 0.000 sec.) occurred before the first
             time tick (at  0.011 sec.).
          The last data point (at 39.061 sec.) occurred after the last
             time tick (at 37.011 sec.).

       Output file = PVB.R03
          recorder sensitivity = 1.8100 cm/g.
          There are 24446 points in the trace.
          Trace length = 39.101 seconds.
          Mean value =  0.24200E+01 cm/sec**2.
          Max. value =  0.30718E+03 at  7.153 seconds.
          Min. value = -0.24064E+03 at  3.681 seconds.
          There are  41 back-stepping points, the largest of which is for
             0.00040 seconds at time =  4.166 seconds.
          The largest difference between two consecutive, unsmoothed
             time-tick intervals was  -2.981% at tick number 49.
          There was a  -1.230% difference betwen the first ( 0.4961 cm.)
             and last ( 0.5022 cm) unsmoothed tick intervals.
          The first data point (at 0.000 sec.) occurred before the first
             time tick (at  0.034 sec.).
          The last data point (at 39.101 sec.) occurred after the last
             time tick (at 37.034 sec.).
    $ VTSP PVB.R03,,,20.0,0.3

       TSPLOT.
       Time series plotting program for AGRAM data,
       24apr84 version.

       [Here follow prompts from the VIEWER device-independent  software:


       (run messages from the plotting software were removed here.)


       end of VIEWER s prompts.]
```


PVB.RD3
304.8
SCALED INSTRUMENT RESPONSE CM/SEC/SEC
310.
-310.
2
4
6
8
10
12
14
16
18
20
(CONTINUED)
22
24
26
28
30
32
34
36
38
40
SECONDS

```
$!
$!
$!   Run HIFRIC for the third trace to get interpolated,
$!   high-cut filtered and instrument-corrected data.
$!   Plot results.
$!
$!
$ HIFRIC PVB.G03=PVB.R03,.039,.60

    HIFRIC, 13sep83 version.

    Input file  = PVB.R03

    Output file = PVB.G03

    Instrument period = 0.039 seconds,
    instrument damping= 0.600 of critical damping.

    Instrument correction and high-cut filter performed on densely
       interpolated data at 600.0 samples per second.
    Transition band for the high-cut filter =  50.00 to 100.00 hz.
    Corrected, filtered data have been decimated by 1/3 to 200.0 samples
       per second.

    There are  7811 points in the trace.
       First value = -0.53025E+01 at  0.000 seconds.
       Last value  = -0.83251E+00 at 39.050 seconds.
       Max. value  =  0.30553E+03 at  7.145 seconds.
       Min. value  = -0.24290E+03 at  3.675 seconds.
$ VTSP PVB.G03,,,,0.3

    TSPLOT.
    Time series plotting program for AGRAM data,
    24apr84 version.


    (run messages from the plotting software were removed here.)
```


PVB.G03
305.5
(CONTINUED)
SECONDS
ACCELERATION CM/SEC/SEC
310.
-310.
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40

```
$!
$!
$!   Compare the effects of the interpolation, high-cut
$!   filter and instrument correction applied in HIFRIC to
$!   uninterpolated, uncorrected data produced by SCALE.
$!   Also compare these to interpolated but uncorrected data.
$!   To do so, rerun HIFRIC with interpolation only, no
$!   instrument correction or filter, then plot two seconds of
$!     (1) uncorrected (SCALE) data,
$!     (2) interplolated (HIFRIC with INTERP) data, and
$!     (3) high-cut filtered, instrument corrected (standard
$!         HIFRIC) data
$!   with an exaggerated scale to illustrate how the  peak value
$!   was affected by processing.
$!
$!
$ HIFRIC PVB.I03=PVB.R03,INTERP

    HIFRIC, 13sep83 version.

    Input file  = PVB.R03

    Output file = PVB.I03

    Interpolate to 200.0 samples per second without instrument correction.

    There are  7821 points in the trace.
       First value = -0.25543E+01 at  0.000 seconds.
       Last value  =  0.20953E+00 at 39.100 seconds.
       Max. value  =  0.30110E+03 at  7.150 seconds.
       Min. value  = -0.24289E+03 at  7.435 seconds.
$ VTSP PVB.R03,.I03,.G03, 6.0,8.0,2.0

    TSPLOT.
    Time series plotting program for AGRAM data,
    24apr84 version.


    (run messages from the plotting software were removed here.)
```

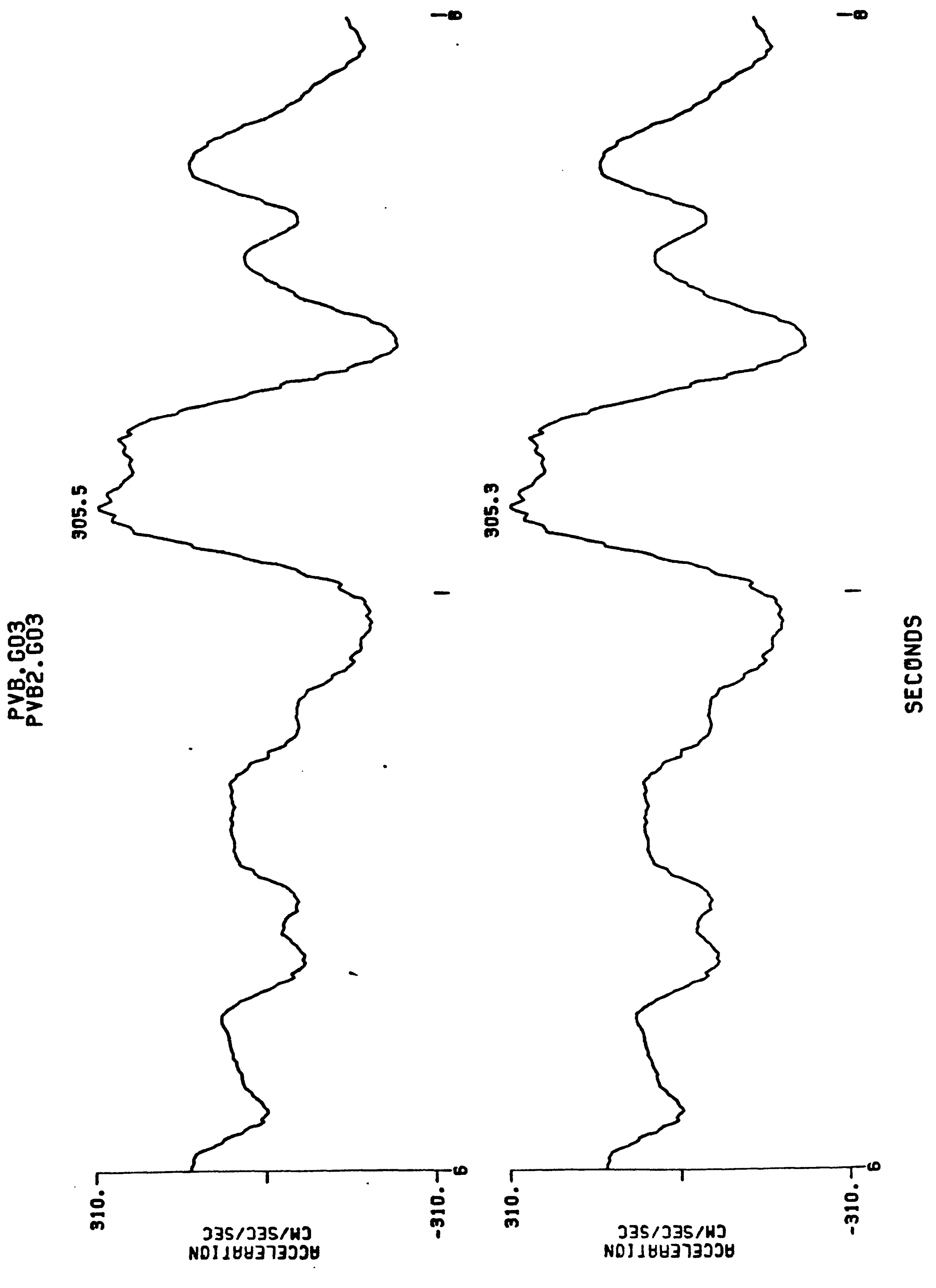

PVB.R03
PVB.I03
PVB.G03
SCALED INSTRUMENT RESPONSE CM/SEC/SEC
SCALED INSTRUMENT RESPONSE CM/SEC/SEC
ACCELERATION CM/SEC/SEC
310.
-310.
304.8
301.1
305.5
6
8
SECONDS

```
$!
$!
$!   The third curve in the plot above shows that the high frequency
$!   digitization noise in the original data near the peak located at
$!   about 7.2 seconds has been amplified by the instrument correction.
$!
$!   The BBFILE program can be used to examine the contents of
$!   the data files near the peak at 7.2 seconds.  The data at 7.2
$!   seconds is in block 12 of the HIFRIC output file and, if
$!   approximately 600 samples per centimeter were digitized, the
$!   data at 7.2 seconds would be in or near block 68 of the SCALE
$!   output file, and in this case it is in block 69.
$!   (block 12 is determined from 11.2 = 7.2 seconds * 200 samples
$!    per second / 128 samples per block.)
$!   (block 68 is determined from 67.5 = 7.2 seconds * 600
$!    samples per second * 2 items (x and y) per sample
$!    / 128 items per block.)
$!
$!
$ BBFILE PVB.G03,12
    PVB.G03 has
           1 integer header blocks, using  -32768 as "undefined",
           1 real header blocks, using -0.30000E-38 as "undefined",
           0 text header blocks,
    and   62 data blocks, each containing 128 real
             acceleration values.


    Integer header block:
    location   content        meaning
    ========   =======        =======
       1           0      number of integer headers, less one, in the file
       2           0      number of text headers in the file
       4           1      =1 if real data, undefined if integer data
      31          62      number of data blocks.
      32           3      index of the last data point within the last block
      41          90      vertical orientation, 0 to 180 from up.
      43          AC      type of data (BU, IR, AC, VL or DP)


    Real header block:
    location   content        meaning
    ========   =======        =======
       1     0.00000E+00  number of real headers, less one.
       5     0.20000E+03  sampling rate in samples per second
      46     0.10000E+05  digitizer output in digitization units/cm.
      47     0.50000E+02  corner frequency used in anti alias filter
      48     0.50000E+02  rolloff bandwith used in anti alias filter
      49     0.39000E-01  instrument period in seconds
      50     0.60000E+00  instrument damping
      51     0.18100E+01  SMA recorder sensitivity or DR100 coil constant
      90     0.00000E+00  time of the first data point
      91    -0.53025E+01  value of the first data point
      92     0.39050E+02  time of the last data point
```

```
  93   -0.83251E+00   value of the last data point
  94    0.71450E+01   time of the max. value
  95    0.30553E+03   max. value
  96    0.36750E+01   time of the min. value
  97   -0.24290E+03   min. value
  98    0.24200E+01   data offset for SCALE data
  99    0.00000E+00   begin time for CORAVD llsqf baseline correction
 100    0.39050E+02   end   time for CORAVD llsqf baseline correction
 101    0.17679E+00   slope of CORAVD linear baseline correction
 102   -0.93284E+01   intercept of CORAVD linear baseline correction
 103    0.21000E+01   source of data: 1.0,2.1,2.2,3.0,3.1, or 3.2


Data block 12:
  -0.39271E+02  -0.23956E+02   0.12731E+02   0.38149E+02   0.46926E+02
   0.68068E+02   0.10091E+03   0.13103E+03   0.13574E+03   0.16160E+03
   0.18227E+03   0.20691E+03   0.23956E+03   0.24343E+03   0.24654E+03
   0.26121E+03   0.28027E+03   0.27987E+03   0.27405E+03   0.29070E+03
   0.30360E+03   0.30553E+03   0.29247E+03   0.28119E+03   0.28705E+03
   0.29014E+03   0.28807E+03   0.27668E+03   0.26627E+03   0.26168E+03
   0.25767E+03   0.25026E+03   0.24294E+03   0.24163E+03   0.24413E+03
   0.24717E+03   0.24786E+03   0.24434E+03   0.24684E+03   0.25670E+03
   0.25998E+03   0.25619E+03   0.25673E+03   0.26219E+03   0.26959E+03
   0.25791E+03   0.24492E+03   0.25162E+03   0.24450E+03   0.23297E+03
   0.22071E+03   0.21149E+03   0.18988E+03   0.16206E+03   0.15838E+03
   0.13833E+03   0.10865E+03   0.89792E+02   0.63195E+02   0.40518E+02
   0.26481E+02   0.64937E+01  -0.27403E+02  -0.27947E+02  -0.37826E+02
  -0.94542E+02  -0.95331E+02  -0.11084E+03  -0.15074E+03  -0.15273E+03
  -0.17189E+03  -0.19583E+03  -0.20108E+03  -0.21203E+03  -0.22774E+03
  -0.23478E+03  -0.24225E+03  -0.24129E+03  -0.23945E+03  -0.24099E+03
  -0.23802E+03  -0.23337E+03  -0.22737E+03  -0.22326E+03  -0.21421E+03
  -0.19516E+03  -0.18722E+03  -0.17122E+03  -0.14426E+03  -0.13023E+03
  -0.11944E+03  -0.10179E+03  -0.81970E+02  -0.66320E+02  -0.54518E+02
  -0.53669E+02  -0.40516E+02  -0.24947E+02  -0.20767E+02  -0.67628E+01
   0.19303E+01   0.82949E+01   0.20926E+02   0.28242E+02   0.33268E+02
   0.36470E+02   0.37381E+02   0.37290E+02   0.36574E+02   0.32801E+02
   0.26939E+02   0.17812E+02   0.57294E+01  -0.69164E+01  -0.15619E+02
  -0.27109E+02  -0.37929E+02  -0.46830E+02  -0.57900E+02  -0.59392E+02
  -0.58914E+02  -0.58368E+02  -0.56346E+02  -0.48822E+02  -0.37894E+02
  -0.19646E+02   0.11070E+01   0.83010E+01
$ BBFILE PVB.R03,68-69
   PVB.R03 has
          1 integer header blocks, using  -32768 as "undefined",
          1 real header blocks, using -0.30000E-38 as "undefined",
          0 text header blocks,
   and  382 data blocks, each containing 128 real
            instrument response values.


   Integer header block:
   location   content        meaning
   ========   =======        =======
      1          0       number of integer headers, less one, in the file
      2          0       number of text headers in the file
  4          1      =1 if real data, undefined if integer data
 31        382      number of data blocks.
 32        124      index of the last data point within the last block
 41         90      vertical orientation, 0 to 180 from up.
 43         IR      type of data (BU, IR, AC, VL or DP)


Real header block:
location   content          meaning
========   =======          =======
    1    0.00000E+00   number of real headers, less one.
   46    0.10000E+05   digitizer output in digitization units/cm.
   51    0.18100E+01   SMA recorder sensitivity or DR100 coil constant
   90    0.00000E+00   time of the first data point
   91   -0.13429E+00   value of the first data point
   92    0.39101E+02   time of the last data point
   93    0.26345E+01   value of the last data point
   94    0.71526E+01   time of the max. value
   95    0.30718E+03   max. value
   96    0.36813E+01   time of the min. value
   97   -0.24064E+03   min. value
   98    0.24200E+01   data offset for SCALE data
  103    0.10000E+01   source of data: 1.0,2.1,2.2,3.0,3.1, or 3.2


Data block 68:
   0.69583E+01  -0.18795E+03   0.69599E+01  -0.18740E+03   0.69615E+01
  -0.18646E+03   0.69630E+01  -0.18649E+03   0.69646E+01  -0.18631E+03
   0.69662E+01  -0.18614E+03   0.69677E+01  -0.18656E+03   0.69693E+01
  -0.18727E+03   0.69708E+01  -0.18692E+03   0.69724E+01  -0.18646E+03
   0.69740E+01  -0.18639E+03   0.69755E+01  -0.18554E+03   0.69771E+01
  -0.18557E+03   0.69786E+01  -0.18442E+03   0.69801E+01  -0.18300E+03
   0.69817E+01  -0.18179E+03   0.69832E+01  -0.18057E+03   0.69848E+01
  -0.17984E+03   0.69865E+01  -0.17918E+03   0.69881E+01  -0.17837E+03
   0.69896E+01  -0.17717E+03   0.69911E+01  -0.17642E+03   0.69927E+01
  -0.17434E+03   0.69936E+01  -0.16878E+03   0.69947E+01  -0.16552E+03
   0.69966E+01  -0.16120E+03   0.69987E+01  -0.16098E+03   0.70003E+01
  -0.15680E+03   0.70011E+01  -0.15266E+03   0.70046E+01  -0.14417E+03
   0.70060E+01  -0.13989E+03   0.70089E+01  -0.13543E+03   0.70110E+01
  -0.13308E+03   0.70131E+01  -0.13162E+03   0.70148E+01  -0.13127E+03
   0.70164E+01  -0.12979E+03   0.70179E+01  -0.12894E+03   0.70196E+01
  -0.12846E+03   0.70210E+01  -0.12724E+03   0.70225E+01  -0.12433E+03
   0.70238E+01  -0.12191E+03   0.70249E+01  -0.11678E+03   0.70257E+01
  -0.11145E+03   0.70275E+01  -0.10692E+03   0.70304E+01  -0.10000E+03
   0.70310E+01  -0.85889E+02   0.70319E+01  -0.85036E+02   0.70350E+01
  -0.74545E+02   0.70358E+01  -0.71529E+02   0.70384E+01  -0.63791E+02
   0.70390E+01  -0.58386E+02   0.70416E+01  -0.53781E+02   0.70434E+01
  -0.48643E+02   0.70449E+01  -0.43903E+02   0.70460E+01  -0.37120E+02
   0.70479E+01  -0.33357E+02   0.70497E+01  -0.27266E+02   0.70514E+01
  -0.23821E+02   0.70528E+01  -0.17175E+02   0.70545E+01  -0.14051E+02
   0.70547E+01  -0.35687E+01   0.70557E+01   0.58244E+00   0.70570E+01
   0.11791E+02   0.70591E+01   0.14600E+02
```

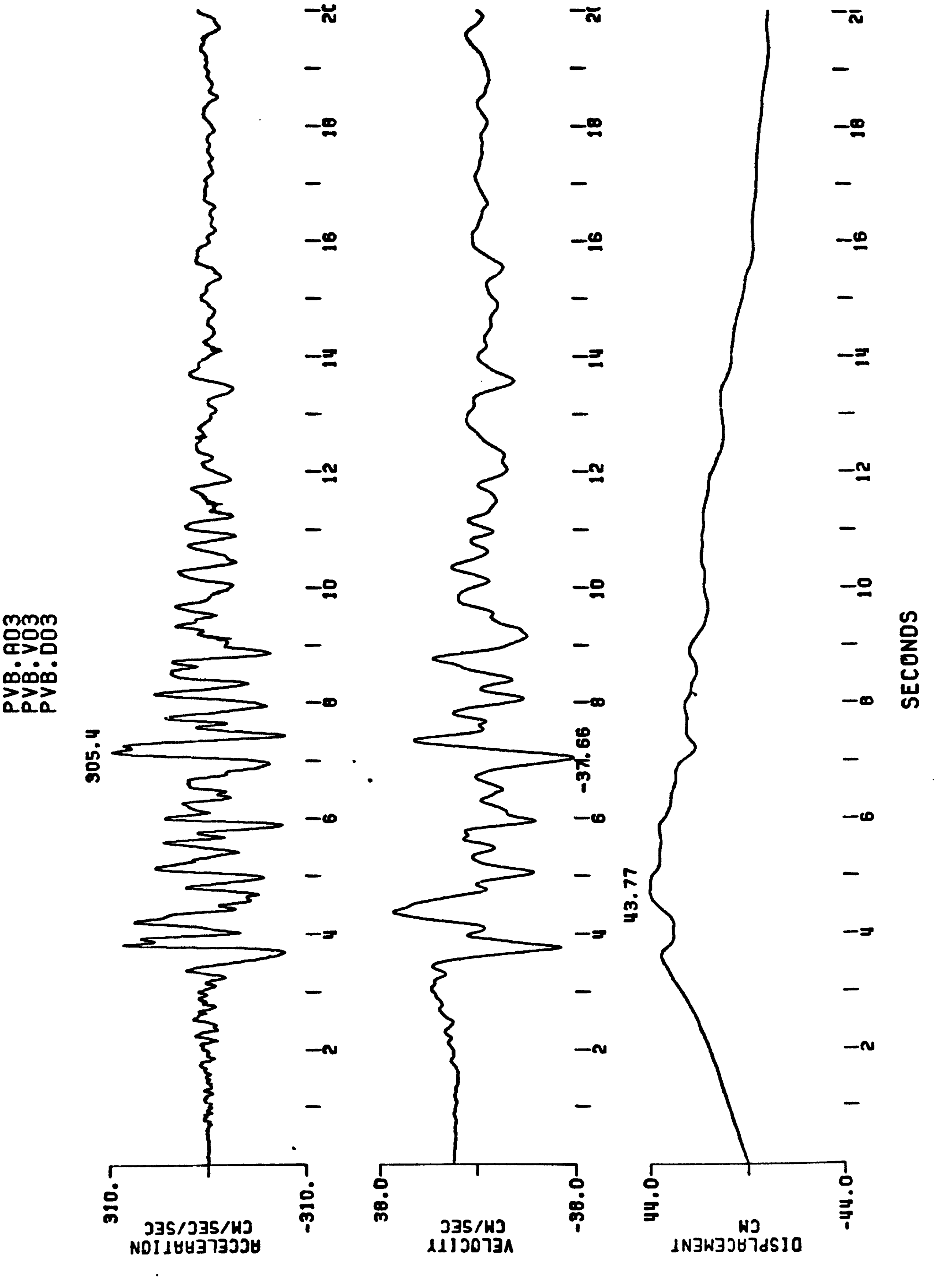
```
Data block 69:
    0.70602E+01   0.26277E+02   0.70623E+01   0.27722E+02   0.70639E+01
    0.39599E+02   0.70648E+01   0.42054E+02   0.70667E+01   0.53398E+02
    0.70680E+01   0.55806E+02   0.70700E+01   0.65894E+02   0.70713E+01
    0.69561E+02   0.70743E+01   0.75246E+02   0.70748E+01   0.81264E+02
    0.70745E+01   0.92185E+02   0.70767E+01   0.96573E+02   0.70787E+01
    0.10700E+03   0.70798E+01   0.11234E+03   0.70813E+01   0.12231E+03
    0.70819E+01   0.12895E+03   0.70850E+01   0.13313E+03   0.70871E+01
    0.13450E+03   0.70870E+01   0.14638E+03   0.70877E+01   0.14920E+03
    0.70903E+01   0.15863E+03   0.70912E+01   0.16362E+03   0.70943E+01
    0.16911E+03   0.70955E+01   0.17445E+03   0.70965E+01   0.18163E+03
    0.70986E+01   0.18567E+03   0.70991E+01   0.20021E+03   0.71005E+01
    0.20457E+03   0.71035E+01   0.21096E+03   0.71042E+01   0.21479E+03
    0.71050E+01   0.22338E+03   0.71061E+01   0.22805E+03   0.71079E+01
    0.23595E+03   0.71095E+01   0.24063E+03   0.71121E+01   0.24582E+03
    0.71139E+01   0.24990E+03   0.71156E+01   0.25432E+03   0.71177E+01
    0.25654E+03   0.71177E+01   0.25654E+03   0.71177E+01   0.25654E+03
    0.71212E+01   0.26399E+03   0.71241E+01   0.27076E+03   0.71264E+01
    0.27582E+03   0.71293E+01   0.27989E+03   0.71338E+01 peak 0.28356E+03
    0.71346E+01   0.28015E+03   0.71378E+01   0.28483E+03   0.71426E+01
    0.29318E+03   0.71467E+01   0.29879E+03   0.71526E+01 ->[0.30718E+03]
    0.71568E+01   0.29815E+03   0.71623E+01   0.29342E+03   0.71647E+01
    0.28928E+03   0.71700E+01   0.28893E+03   0.71764E+01   0.28860E+03
    0.71795E+01   0.28443E+03   0.71823E+01   0.28074E+03   0.71863E+01
    0.27520E+03   0.71890E+01   0.26919E+03   0.71927E+01   0.26504E+03
    0.71959E+01   0.26110E+03   0.71959E+01   0.26110E+03   0.71959E+01
    0.26110E+03   0.71999E+01   0.25530E+03
```

```
$!
$!
$!   Notice that the frequency-domain instrument correcting
$!   algorithm has nearly the same effect as the time-domain
$!   algorithm.
$!
$!
$ HIFRIC PVB2.G03=PVB.R03,.039,.60,FDIC

    HIFRIC, 13sep83 version.

    Input file  = PVB.R03

    Output file = PVB2.G03

    Instrument period = 0.039 seconds,
    instrument damping= 0.600 of critical damping.

    Instrument correction and high-cut filter performed on densely
       interpolated data at 600.0 samples per second.
    Frequency domain instrument correction method used.
    Transition band for the high-cut filter =  50.00 to 100.00 hz.
    Corrected, filtered data have been decimated by 1/3 to 200.0 samples
       per second.

    There are  7821 points in the trace.
       First value = -0.52977E+01 at  0.000 seconds.
       Last value  = -0.12023E+00 at 39.100 seconds.
       Max. value  =  0.30531E+03 at  7.145 seconds.
       Min. value  = -0.24285E+03 at  3.675 seconds.
$ VTSP PVB.G03,PVB2.G03,  6.0,8.0,2.0

    TSPLOT.
    Time series plotting program for AGRAM data,
    24apr84 version.


    (run messages from the plotting software were removed here.)
```

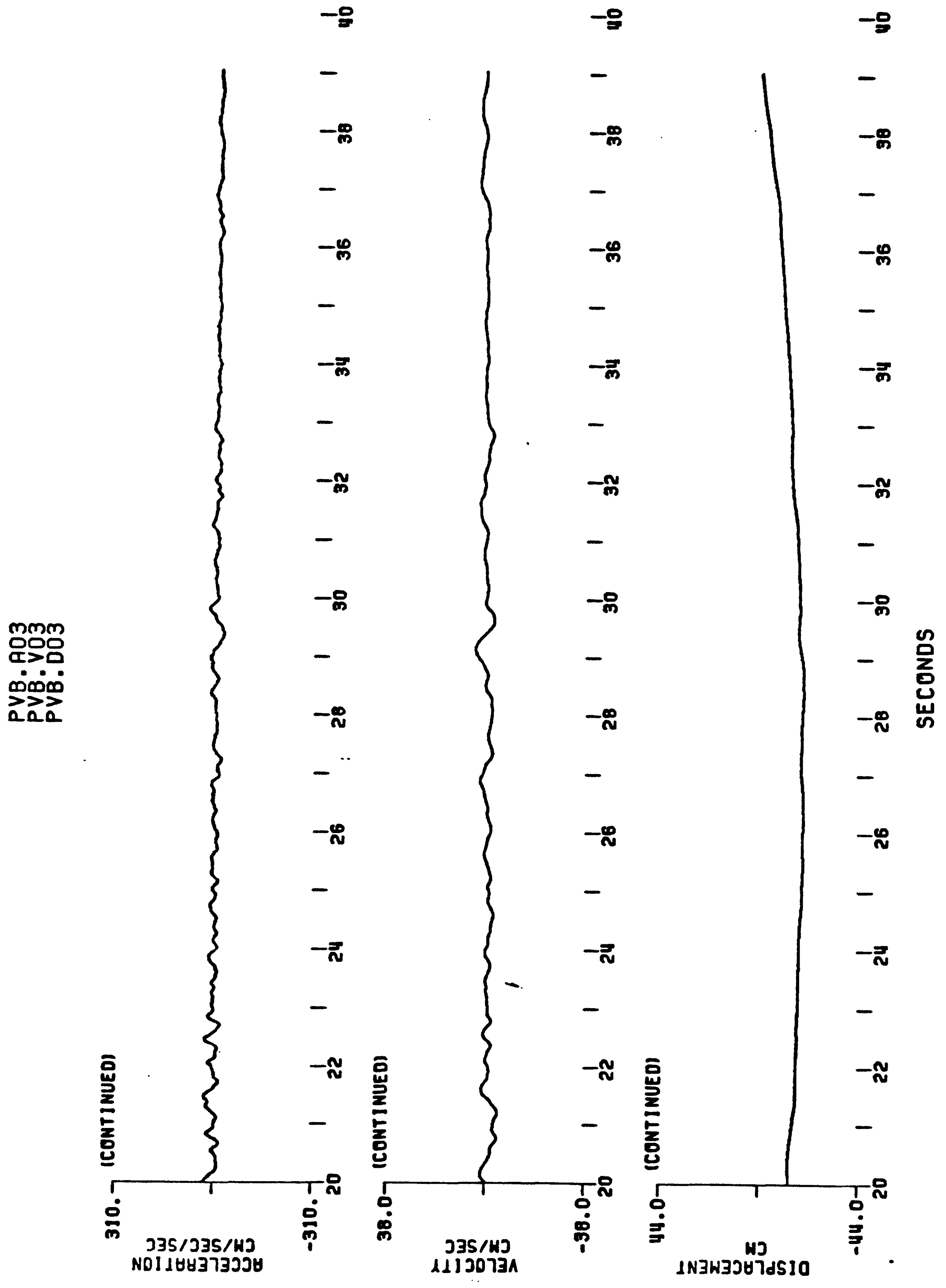

PVB.G03
PVB2.G03
305.5
305.3
310.
-310.
ACCELERATION
CM/SEC/SEC
6
7
8
SECONDS

```
$!
$!
$!   Run CORAVD with linear baseline correction to the velocity
$!   and without long-period filtering.  Plot results.
$!
$!
$ CORAVD PVB.=PVB.G03

    CORAVD, 14dec83 version.
    Integrating and Baseline correcting program for AGRAM data.

    Input file  = PVB.G03

    Output acceleration file= PVB.A03
           velocity file=     PVB.V03
           displacement file= PVB.D03

    Acceleration corrected by subtracting  0.177,
    Velocity corrected by subtracting the line with slope=  0.177
       and intercept= -9.328.
    Slope and intercept were calculated (in HIFRIC) as the linear least
       squares fit to the velocity between  0.000 and 39.050 seconds.

    No filtering performed.

    There are  7811 data points in the acceleration file.
       First value = -0.54792E+01 at  0.000 seconds.
       Last value  = -0.10093E+01 at 39.050 seconds.
       Max. value  =  0.30535E+03 at  7.145 seconds.
       Min. value  = -0.24308E+03 at  3.675 seconds.

    There are  7811 data points in the velocity file.
       First value =  0.93151E+01 at  0.000 seconds.
       Last value  =  0.24444E+01 at 39.050 seconds.
       Max. value  =  0.32774E+02 at  4.370 seconds.
       Min. value  = -0.37658E+02 at  7.050 seconds.

    There are  7811 data points in the displacement file.
       First value =  0.23288E-01 at  0.000 seconds.
       Last value  = -0.61151E-02 at 39.050 seconds.
       Max. value  =  0.43774E+02 at  4.700 seconds.
       Min. value  = -0.19732E+02 at 28.390 seconds.
$ VTSP PVB.A03,.V03,.D03

    TSPLOT.
    Time series plotting program for AGRAM data,
    24apr84 version.


    (run messages from the plotting software were removed here.)
```

B-29
Examples
CORAVD/TSPLOT



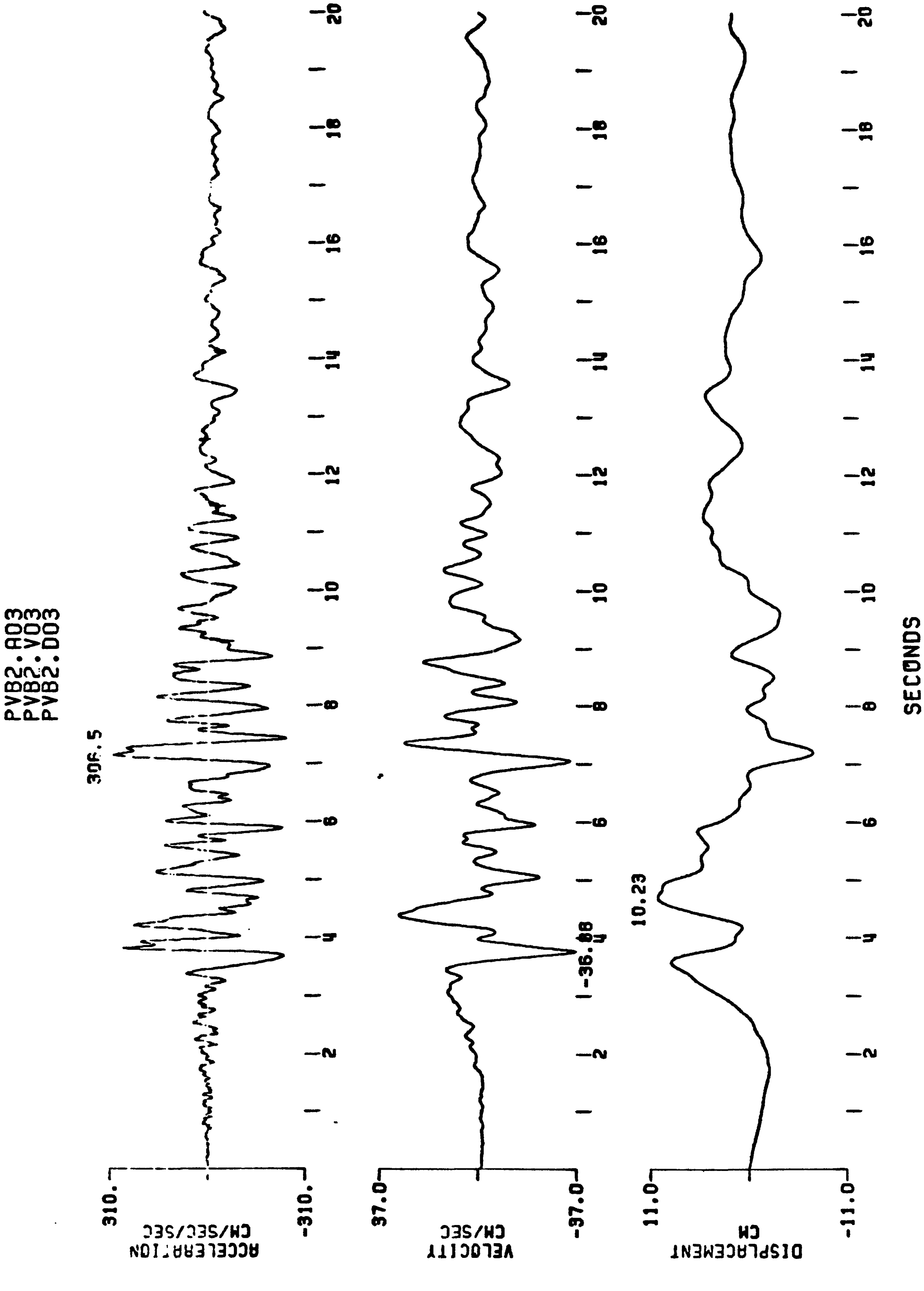
B-30
Examples
CORAVD/TSPLOT

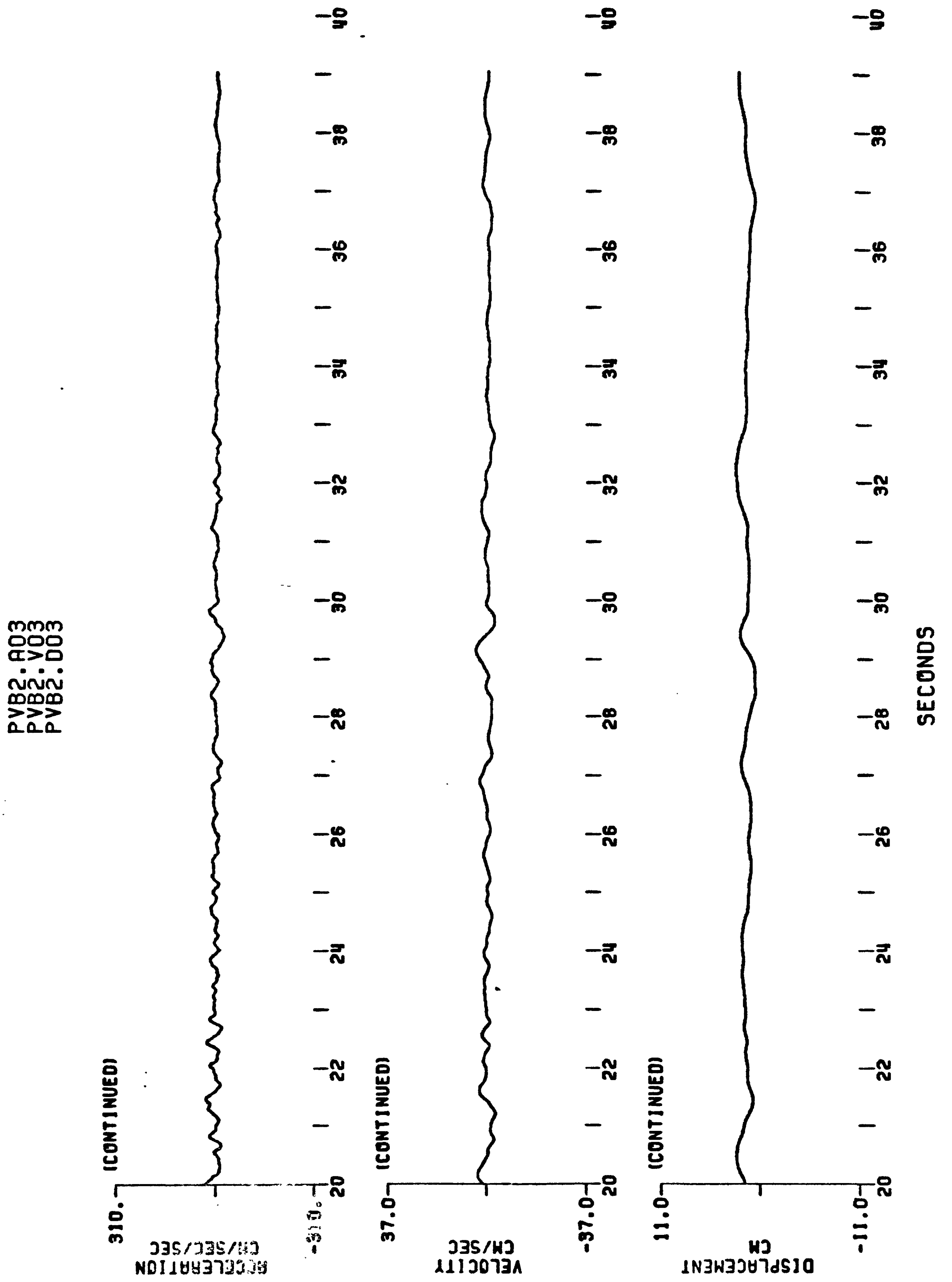


```
$!
$!
$!   Run CORAVD with long-period filtering but no linear
$!   baseline correction and plot results.
$!
$!
$ CORAVD PVB2.=PVB.G03,0.0,0.0,BI,0.100,2

    CORAVD, 14dec83 version.
    Integrating and Baseline correcting program for AGRAM data.

    Input file  = PVB.G03

    Output acceleration file= PVB2.A03
           velocity file=     PVB2.V03
           displacement file= PVB?.D03

    No linear baseline correction performed.

    Velocity filtered with
       bidirectional, high-pass Butterworth filter,
       corner frequency =  0.100 cps, and rolloff order = 2
       The data was padded during the filtering process
       with  3905 trailing points
       containing zeros.

    Acceleration filtered with
       bidirectional, high-pass Butterworth filter,
       corner frequency =  0.100 cps, and rolloff order = 2
       The data was padded during the filtering process
       with  3905 trailing points
       containing zeros.

    There are  7811 data points in the acceleration file.
       First value = -0.51691E+01 at  0.000 seconds.
       Last value  = -0.94471E+00 at 39.050 seconds.
       Max. value  =  0.30654E+03 at  7.145 seconds.
       Min. value  = -0.24147E+03 at  7.420 seconds.

    There are  7811 data points in the velocity file.
       First value = -0.10361E+01 at  0.000 seconds.
       Last value  = -0.18842E+00 at 39.050 seconds.
       Max. value  =  0.30160E+02 at  4.375 seconds.
       Min. value  = -0.36883E+02 at  3.755 seconds.

    There are  7811 data points in the displacement file.
       First value = -0.25903E-02 at  0.000 seconds.
       Last value  =  0.20439E+01 at 39.050 seconds.
       Max. value  =  0.10226E+02 at  4.690 seconds.
       Min. value  = -0.72372E+01 at  7.205 seconds.
```

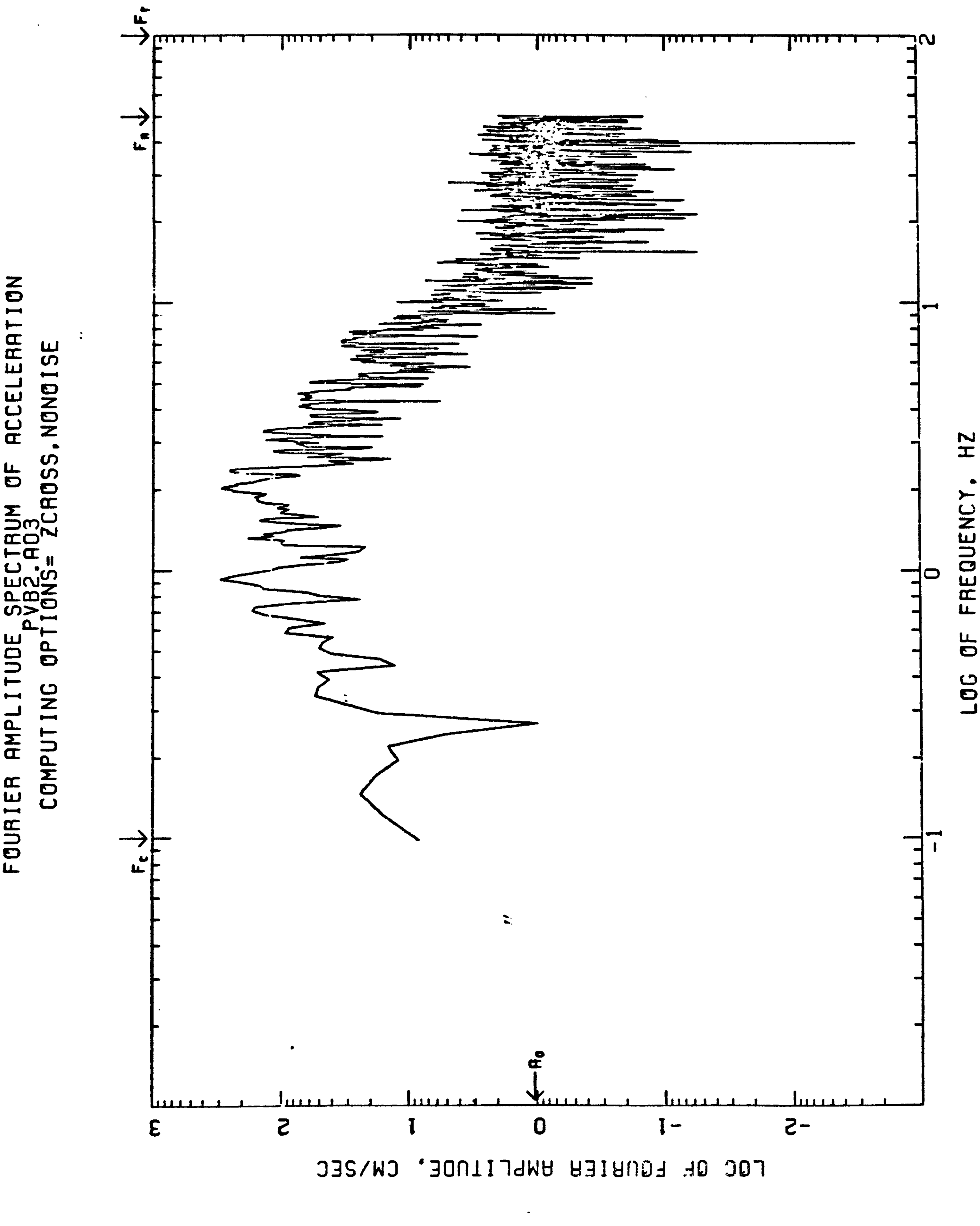
```
$ VTSP PVB2.A03,.V03,.D03

    TSPLOT.
    Time series plotting program for AGRAM data,
    24apr84 version.


    (run messages from the plotting software were removed here.)
```

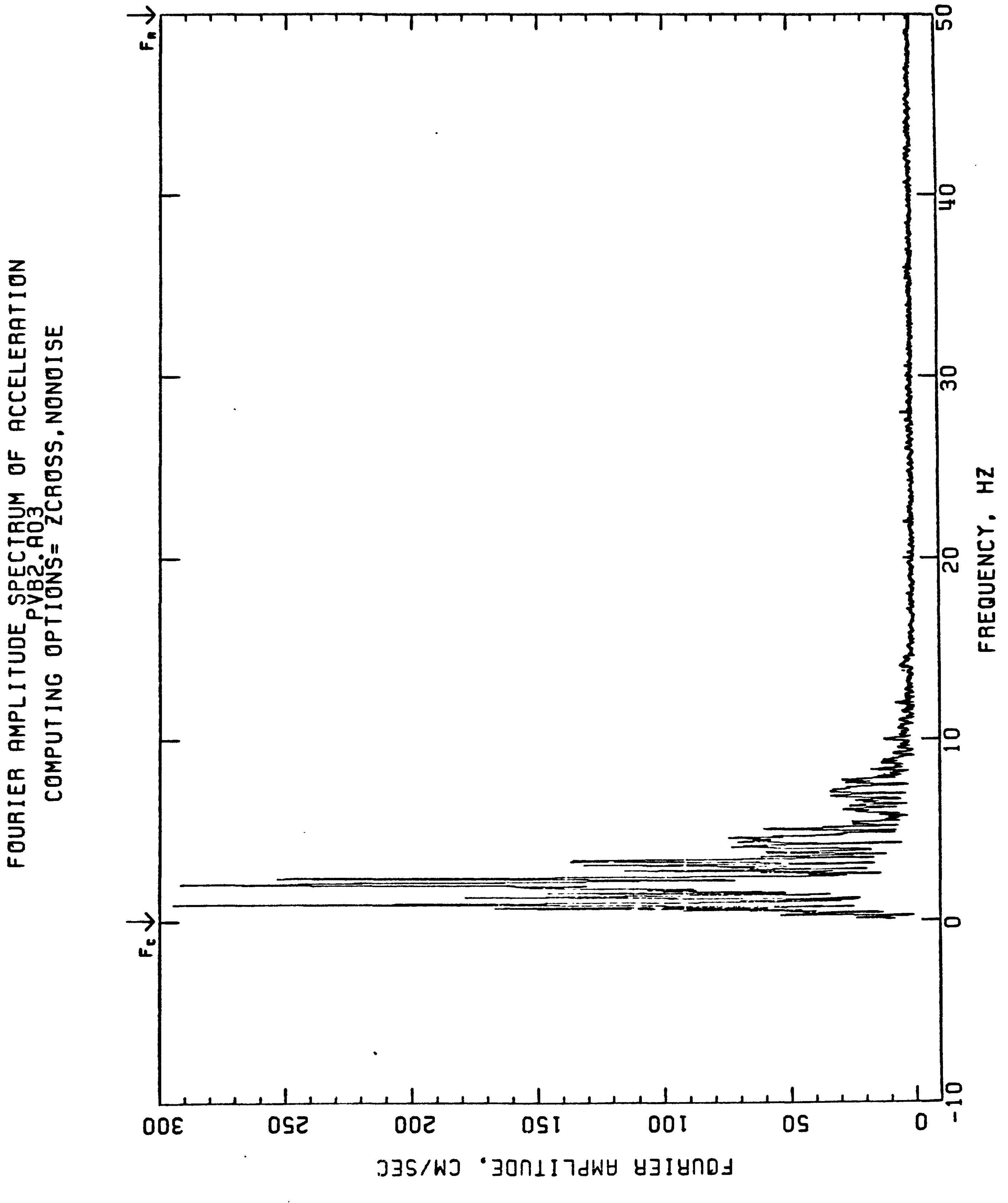
B-33
Examples
CORAVD/TSPLOT




PVB2.A03
PVB2.V03
PVB2.D03
(CONTINUED)
310.
-310.
ACCELERATION
CM/SEC/SEC
37.0
-37.0
VELOCITY
CM/SEC
11.0
-11.0
DISPLACEMENT
CM
20
22
24
26
28
30
32
34
36
38
40
SECONDS

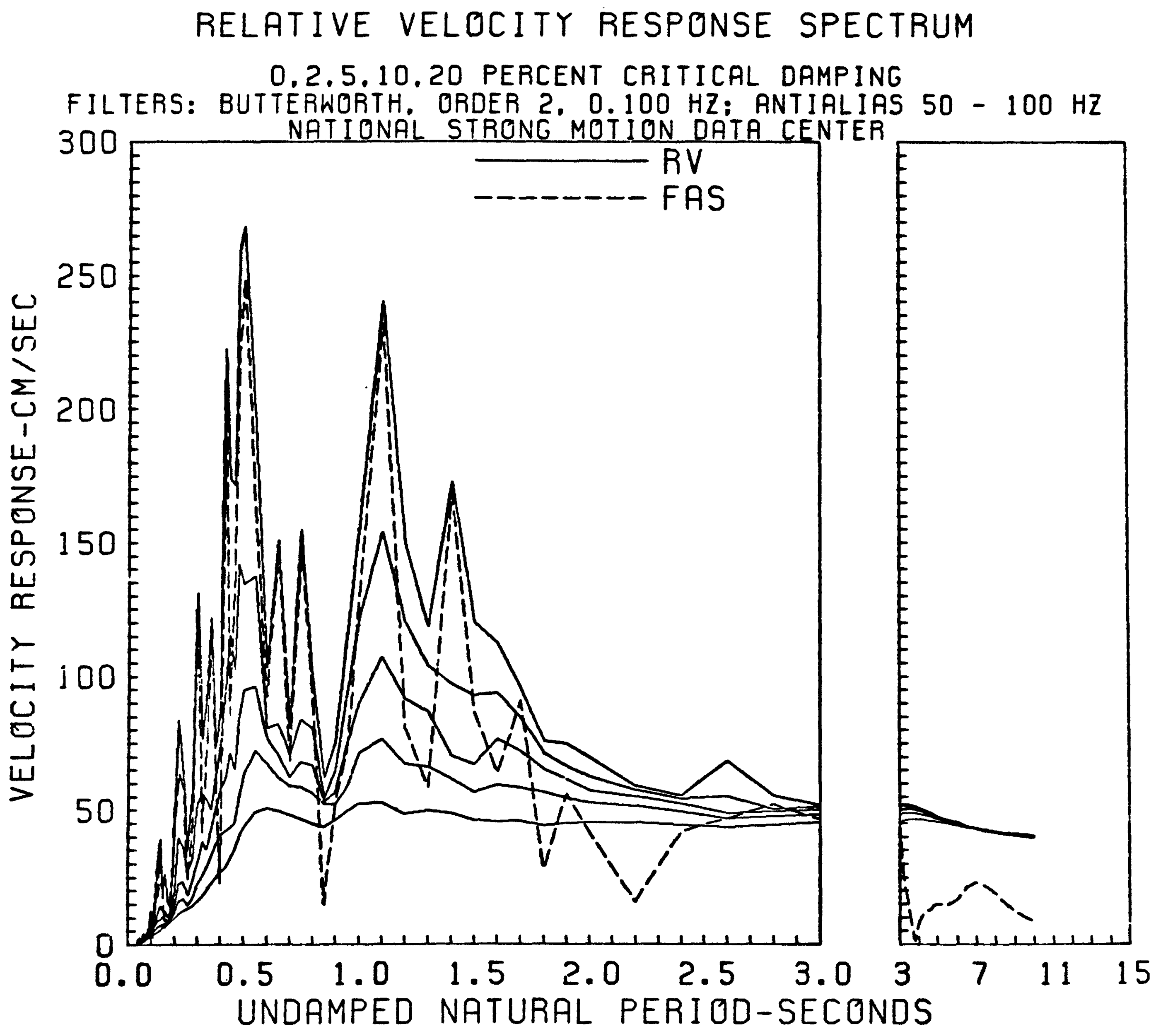
```
$!
$!   Run FASPLT to calculate and plot Fourier amplitude
$!   spectrum.
$!
$ VFAS PVB2.A03,ZCROSS,NONOISE,LOGLOG,LINLIN

    FASPLT.
    Fourier Spectra plotting program for AGRAM data,
    17apr84 version.

    [Here follow prompts from the VIEWER device-independent  software:


    (run messages from the plotting software were removed here.)


    end of VIEWER s prompts.]

    The time series data has been padded with   381 trailing zeros.
       The data samples preceding the first zero crossing (    21 points)
       and the data samples following the last zero crossing (     0
       points) were also reset to zero.
    The   8192 time-series samples have been transformed to  4097
       complex samples in the frequency domain.
    Amplitude at zero frequency = 0.10375E+01

    Wait 5 seconds here to allow user to see previous run messages


    (run messages from the plotting software were removed here.)
```


FOURIER AMPLITUDE SPECTRUM OF ACCELERATION
PVB2.A03
COMPUTING OPTIONS= ZCROSS,NONOISE
LOG OF FREQUENCY, HZ
LOG OF FOURIER AMPLITUDE, CM/SEC
-1
0
1
2
3
2
1
0
-1
-2
Fc
Fn
Fr
Ao


FOURIER AMPLITUDE SPECTRUM OF ACCELERATION
PVB2.A03
COMPUTING OPTIONS= ZCROSS,NONOISE
FOURIER AMPLITUDE, CM/SEC
FREQUENCY, HZ
F_c
F_n
0
50
100
150
200
250
300
-10
0
10
20
30
40
50

```
$!
$!
$!   Run PHASE3 to calculate relative response spectra, then
$!   run PH3PLT to plot results.
$!
$!
$ COPY PVB2.A03 PVB2.A01  ! phase3 expects to process 3 files
$ COPY PVB2.A03 PVB2.A02  !    "
$ COPY PUB1:[AGRAM.TESTCASES]NOTITLE.TXT PVB2.TIT
$ PHASE3
  This is program Phase3.
  Enter Phase2 output file to be processed by Phase3:
  (If Faze2b output files, enter prefix only.)

  ENTER NAME OF OUTPUT FILE.

    8    0    0    0 PVB2.PH3
  DIAGNOSTIC FILE=DIAGN.PH3
  CALCULATING SPECTRA WITH CRIT. DAMPING =0.00
  CALCULATING SPECTRA WITH CRIT. DAMPING =0.02
  CALCULATING SPECTRA WITH CRIT. DAMPING =0.05
  CALCULATING SPECTRA WITH CRIT. DAMPING =0.10
  CALCULATING SPECTRA WITH CRIT. DAMPING =0.20
  FINISHED TRACE NO. 1.
  CALCULATING SPECTRA WITH CRIT. DAMPING =0.00
  CALCULATING SPECTRA WITH CRIT. DAMPING =0.02
  CALCULATING SPECTRA WITH CRIT. DAMPING =0.05
  CALCULATING SPECTRA WITH CRIT. DAMPING =0.10
  CALCULATING SPECTRA WITH CRIT. DAMPING =0.20
  FINISHED TRACE NO. 2.
  CALCULATING SPECTRA WITH CRIT. DAMPING =0.00
  CALCULATING SPECTRA WITH CRIT. DAMPING =0.02
  CALCULATING SPECTRA WITH CRIT. DAMPING =0.05
  CALCULATING SPECTRA WITH CRIT. DAMPING =0.10
  CALCULATING SPECTRA WITH CRIT. DAMPING =0.20
  FINISHED TRACE NO. 3.
FORTRAN STOP
$ PH3PLT
  THIS IS PROGRAM PH3PLT.
  ENTER PHASE3 OUTPUT FILE TO BE PLOTTED BY PH3PLT:


    (run messages from the plotting software were removed here.)


  FINISHED TRACE NO. 1.
  FINISHED TRACE NO. 2.
  FINISHED TRACE NO. 3.
FORTRAN STOP
$!
$!   Now run VIEWERR and RASM to get a hard-copy plot from the
$!   batch.plt file generated by PH3PLT.
$!
$ RUN PUB1:[BGMF]VIEWERR
```

```
    (run messages from the plotting software were removed here.)
$ MCR RASM
    (run messages from the plotting software were removed here.)
$ DELETE *.PLV;*
```

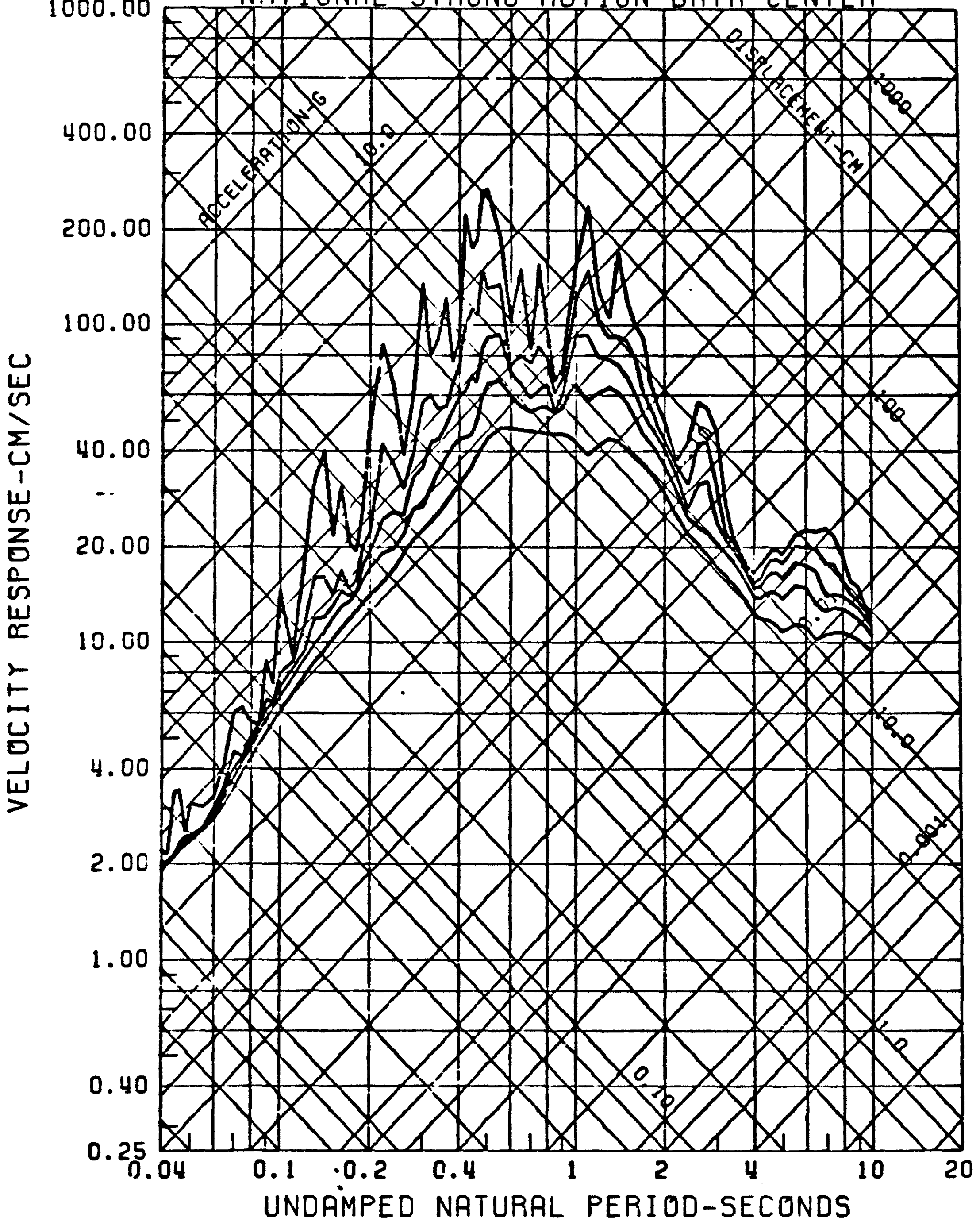

RELATIVE VELOCITY RESPONSE SPECTRUM
0,2,5,10,20 PERCENT CRITICAL DAMPING
FILTERS: BUTTERWORTH, ORDER 2, 0.100 HZ; ANTIALIAS 50 - 100 HZ
NATIONAL STRONG MOTION DATA CENTER
RV
FAS
VELOCITY RESPONSE-CM/SEC
0
50
100
150
200
250
300
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3
7
11
15
UNDAMPED NATURAL PERIOD-SECONDS

RESPONSE SPECTRA

0,2,5,10,20 PERCENT CRITICAL DAMPING

FILTERS: BUTTERWORTH, ORDER 2, 0.100 HZ: ANTIALIAS 50 - 100 HZ

NATIONAL STRONG MOTION DATA CENTER



```
$!
$!
$!   How many data files were generated?  They should either
$!   be archived or deleted.
$!
$!
$ dir PVB.*,PVB2.*

Directory PUB1:[CONVERSE.TEST]

PVB.A03;1           PVB.BOU;1           PVB.BUT;1           PVB.D03;1
PVB.G03;1           PVB.I03;1           PVB.R01;1           PVB.R02;1
PVB.R03;1           PVB.V03;1           PVB2.A01;1          PVB2.A02;1
PVB2.A03;1          PVB2.D03;1          PVB2.G03;1          PVB2.PH3;1
PVB2.TIT;1          PVB2.V03;1

Total of 18 files.
$!
$!
$!   end of job.
$!
$end: continue
$eoj
  CONVERSE     job terminated at 29-JUN-1984 18:56:15.23

  Accounting information:
  Buffered I/O count:          1875    Peak working set size:   421
  Direct I/O count:           23712    Peak page file size:    1382
  Page faults:                 7295    Mounted volumes:           0
  Elapsed CPU time:     0 00:23:09.51  Elapsed time:     0 01:31:18.33
```

APPENDIX C

Differences between the AGRAM programs and programs used in the past

Some of the differences between the routine processing steps provided by the AGRAM programs and those formerly provided by the PHASE programs are listed in this appendix.

BUTTER

The AGRAM programs operate on data that BUTTER has not decimated so that HIFRIC can make best use of the densely digitized data. The PHASE programs usually processed data on which the decimation option in BUTTER had been applied.

SCALE

PHASE1 removed backsteppping points, SCALE does not.

PHASE1 smoothed the reference trace, SCALE does not.

When a data trace extends beyond the first or last time tick, SCALE uses the length of the nearest (first or last) interval to extend the time ticks. PHASE1 extended the ticks by using the average of all the tick intervals.

HIFRIC

More accurate approximations of the derivatives required by the damped harmonic oscillator equation are used in HIFRIC than the centered-difference method for differentiation that had been used in PHASE2. The standard HIFIRC method incorporates Fourier differentiation (i.e., $\frac{d}{dt} e^{i\omega t} = i\omega e^{i\omega t}$) in convolution operators, which are applied in the time-domain, to approximate the first and second derivatives required. The alternative HIFRIC method applies Fourier differentiation in the frequency domain.

The high-cut filters used in HIFRIC are also different from the Ormsby convolution that had been used in PHASE2. In the standard HIFRIC method, a filter is incorporated in the same convolutions that calculate the derivatives in the time domain. With the alternative frequency-domain method, the spectrum is multiplied by a cosine taper in the transition band where the filter gain decreases from one to zero and is set to zero in the rejection band.

CORAVD

When filtering is used in CORAVD, each time-series is filtered just once. Velocity is calculated from acceleration before acceleration is filtered; displacement is calculated from the filtered velocity and is not refiltered itself. Earlier programs such as PHASE2 had calculated velocity from a filtered version of the acceleration, filtered the velocity, calculated displacement from the filtered velocity and filtered the displacement. This refiltering of a time series determined from an already filtered time series magnifies any distortion inherent in the filter, so CORAVD filters each time series just once, or preferably not at all.

PHASE2 used the Ormsby filtering algorithm but in routine processing CORAVD uses the significantly faster bidirectional Butterworth filter algorithm. Brief discussions of the advantages of the Butterworth filter in addition to its increased speed are given in references [11] and [12].

The table on the following page compares the processing steps once used in PHASE2 to the steps now used in CORAVD. In the table, "llsqf" refers to a linear least-square fit, "acc(t)" refers to acceleration as a function of time, "vel" refers to velocity, "disp" refers to displacement, and y(t) refers to a straight line function. "Option A" is the linear correction option and "option B" is the low-cut filter option.

FASPLT

Rather than padding the input data with trailing zeros as FASPLT does, PHASE4 reinterpolated the data so it would completely fill 2^n points.

PHASE4 did no tapering.

PHASE4 smoothed the amplitudes with a 1/4, 1/2, 1/4 running mean. Although FASPLT will do the same by option, FASPLT does not smooth in the default case.

Step	PHASE2	CORAVD
1	- Interpolate to 100 equally spaced samples per second, filter out high frequency noise, and perform instrument correction.	- No. The counterparts of these calculations are done in HIFRIC now. The interpolation is now at 200 samples per second.
2	- Remove linear trend from acceleration: llsqf [acc(t)] = y(t) = mt + b acc(t) = acc(t) - y(t)	- No. This has already been done in BUTTER.
3	- Calculate velocity from acceleration: $vel(t) = \int acc(t)\, dt$	- Yes.
4	- Fit a straight line to the velocity: llsqf [vel(t)] = y(t) = mt + b	- Yes, with option A.
5	- Subtract the slope of the line fitted to velocity from the acceleration: acc(t) = acc(t) - m	- Yes, with option A.
6	- Subtract a baseline from acceleration: acc(t) = acc(t) - baseline(t) where the baseline is acc(t) processed as follows: . pad both ends of the curve with a mirror image of itself, . smooth with a 0.4-second, equal-weight running average, . decimate, retaining every 10th point, . apply low-pass Ormsby filter.	- Option B filters long periods from acceleration, but uses: . Padding with zeros, . No smoothing, . No decimation, . Butterworth high-pass filter applied directly to acceleration rather than Ormsby low-pass filter applied to a baseline.
7	- Recompute velocity from the corrected acceleration: $vel(t) = \int acc(t)\, dt$	- No.
8	- Fit another straight line to the recomputed velocity: llsqf [vel(t)] = y(t) = mt + b	- No. Already have a fit from step 4 since CORAVD velocity is not recomputed in step 7.
9	- Subtract the linear trend from velocity: vel(t) = vel(t) - y(t)	- Yes, with option A.
10	- Subtract the slope of the line fitted to velocity from the acceleration (a second time!): acc(t) = acc(t) - m	- No.
11	- Subtract a baseline from velocity: vel(t) = vel(t) - baseline(t) where the baseline is determined from velocity as it was for acceleration.	- Option B applies the same filter to velocity as it does to acceleration.
12	- Calculate displacement from velocity: $disp(t) = \int vel(t)\, dt$	- Yes.
13	- Subtract a baseline from displacement: disp(t) = disp(t) - baseline(t) where the baseline is determined from displacement as it was for velocity and acceleration.	- No.

Figure C.1: Compare processing steps in PHASE2 and CORAVD.

Appendix D

Distribution of FORTRAN Code

The FORTRAN code for all the AGRAM programs except the PHASE3 and RSPECT response spectra programs are available on magnetic tape. To acquire information about the most recent version of the code, telephone April Converse at the USGS offices in Menlo Park California, (408) 323-8111, extension 2881, or mail a request to:

April Converse (AGRAM tape)
Branch of Engineering Seismology and Geology
U.S. Geological Survey
345 Middlefield Road, Mail Stop 977
Menlo Park, CA 94025, USA

Your name will be added to the mailing list of those who will be notified whenever new versions of the user's manual (this report) or the distribution tape are available.

The first file on the distribution tape is an introduction and table of contents for the other files on the tape. A print-out of the current version of that first file is given here.

First file on the distribution tape:

This is PUB1:[AGRAM.EXPTAPE]AGRAMTAPE.TOC
01jul84

This tape contains FORTRAN source decks and related files used to construct the AGRAM computer programs at the National Strong-Motion Data Center (NSMDC) of the U.S. Geological Survey (USGS). The programs are used by the USGS for routine processing of analog strong-motion accelerograms. USGS Open-File Report 84-525, titled "AGRAM: a Series of Computer Programs for Processing Digitized Strong-Motion Accelerograms; version 2.0", provides descriptions of the programs and instructions for their use.

The AGRAM programs are not in a final form; they are currently under development and are frequently changed at NSMDC. Many features intended for the programs have yet to be implemented, some of the options they currently provide have yet to be evaluated, and some of the code needs to be revised to conform to standard FORTRAN 77. If support for the development of the programs continues, new versions of this distribution tape will be issued as the programs evolve. If you wish to be notified whenever this distribution tape is updated, send your name and mailing address to the authors.

There are 18 files on the tape. They are:

file number	file name at NSMDC	content
1	agramtape.toc	This text. An introduction and table of contents for the tape.
2	agramhelp.txt	Text for the on-line help facility at NSMDC. Note that the "changes" section in this file lists the changes that have been made to the AGRAM programs since the Open-File report was last issued.
3	progagram.nts	Miscellaneous notes: programming notes, file naming conventions, maintenance notes, identify where non-standard fortran is used, and so forth.
4	iomtap.vax	Program to read an IOM tape and convert it's data from IOM to VAX binary.
5	butter.vax	Frame butting program.
6	reform.vax	Data file reformatting program.
7	scale.vax	SCALE program.
8	hifric.vax	HIFRIC program.
9	coravd.vax	CORAVD program.
10	fasplt.vax	FASPLT program.
11	tsplot.vax	Time-series plotting program.
12	bbfile.vax	Blocked binary file dumping program.
13	bwrite.vax	Data file header changing program.
14	rotate.vax	ROTATE program.
15	smashlib.vax	Subroutine library.
16	sample.com	First sample command deck shown in appendix B of the Open File report.
17	appb.bat	Second sample command deck shown in appendix B.

18	pvb.bou	Sample data file from the BUTTER run shown in appendix B. This is the OUTPUT from a BUTTER run in text (not binary) form.

Each program file on the tape contains a collection of card-image source decks. The beginning of each deck is marked with a separating line that begins with an up-arrow symbol, ends with the name of the deck to follow, and contains a line of dots between the up-arrow and the name. The first deck in a program file is a "workshop" deck that names all the other decks in the file, serving as a table of contents for the file. A program file contains all decks that have anything to do with the program, and since the programs are still in the development stage, some spurious and potentially confusing decks may be included.

These programs require a full FORTRAN 77 compiler, a 9-track tape drive, a plotting device (preferably one driven by Calcomp compatible software), and storage space on a disk. On the VAX machines at NSMDC, 15,000 blocks of disk space are required to store the data files generated while processing one 40-second, 3-component record and 40,000 blocks of disk space are required to store the programs (FORTRAN code, compiled code and other related files). A disk "block", on a VAX, contains 128 thirty-two bit words.

When installing these programs at another site for the first time, it would be best to first install and test the little TSTLIB program from the subroutine library, SMASHLIB. TSTLIB tests many of the subroutines on smashlib. Most importantly, TSTLIB exercises most of the byte manipulating code that may need to be rewritten at other sites. The results from the TSTLIB program should be equivalent to those in the TSTLIB.OUT deck given in the SMASHLIB file. Next, install the REFORM, SCALE, and TSPLOT programs and whatever subroutines from SMASHLIB that these three programs require. Once these three programs have been installed and tested (test using the REFORM, SCALE and TSPLOT commands shown in APPB.BAT with the data given in PVB.BOU), installing all the other programs should be quite simple. Refer to the programming notes given in the third file of this distribution tape for information about machine or site dependent aspects of these programs that may need to be changed to install the programs at another site.

At NSMDC, the AGRAM programs are used on a VAX computer having virtually unlimited address space. At one time, however, most of the programs were installed on a computer (a DEC 11/70) having a 64K byte address space. (8 bits per byte.) If you wish to install the programs on a computer with limited address space, request from the program authors a copy of the commands once used at NSMDC to overlay the code to fit within the 64K bytes. These command decks are now out-of-date since the programs have changed since they were installed on the 64K machine, but even so, the commands may be a help in deciding how to overlay the code at your site.

The program authors are quite willing to answer any questions about the programs by letter or by phone.

April Converse
Peter Mork

U.S. Geological Survey
Mail Stop 978
345 Middlefield Road
Menlo Park, CA 94025

telephone: (415) 323-8111
extension 2881
or FTS 467-2881

References and Bibliography

General

[1] Hudson, D.E. (1979). Reading and Interpreting Strong Motion Accelerograms: Earthquake Engineering Research Institute, 2620 Telegraph Avenue, Berkeley, CA 94704. 112 pages.

(This monograph provides an introduction to strong-motion accelerograms. It describes the computer processing used in the original CalTech data project.)

[2] Hudson, D.E., (1976). "Strong-Motion Earthquake Accelerograms; Index Volume": Earthquake Engineering Research Laboratory report number EERL 76-02, California Institute of Technology, Pasadena, CA.

(This is the final, summarizing report from the CalTech data project.)

[3] "Strong-Motion Earthquake Accelerograms, Digitization and Analysis, 1971 records": USGS Open-File Report Number 76-609, USGS, Menlo Park, CA.

(This report is the first of the series of strong-motion data reports published by the USGS. It summarizes the processing steps used at that time. It also contains a longer bibliography than is given here.)

[4] Brady, A.G.; Perez, V.; and Mork, P.N. (1982). "Digitization and Processing of Main-shock Ground-motion Data from the USGS Accelerograph Network" in "The Imperial Valley, California, Earthquake of October 15, 1979": Geological Survey Professional Paper 1254, pp. 385-406.

(This report gives a detailed description of the decisions made while processing the records recovered from the Imperial Valley earthquake of October 15, 1979.)

[5] Maley, R.; Brady, G.; Etheridge, E.; Johnson, D.; Mork, P.; and Switzer, J. (1983). "Analog Strong Motion Data and Processed Main Event Records Obtained by U.S. Geological Survey Near Coalinga, California" in "The Coalinga Earthquake Sequence Commencing May 2, 1983; (Preliminary report on field investigations and data collected as of May

12, 1983)": USGS Open-File Report Number 83-511; USGS, Menlo Park, CA.

(This is the first published report for which the AGRAM programs were used.)

[6] Trifunac, M.D., and V.W. Lee (1979). "Automatic Digitization and Processing of Strong Motion Accelerograms": Department of Civil Engineering, Report number 79-15 I and II, University of Southern California, Los Angeles, CA.

(These reports describe a system similar to the AGRAM programs but which is used by USC and by CDMG.)

[7] Digital Signal Processing Committee of the IEEE (1979). Programs for Digital Signal Processing: IEEE press, The Institute of Electrical and Electronics Engineers, Inc., New York.

(This book presents discussions and software for Discrete Fast Fourier Transform methods, among others.)

<u>HIFRIC</u>

[8] Raugh, M.R. (1981). "Procedures for Analysis of Strong-motion Records" (abstract): Earthquake Notes, Vol. 52, No. 1.

[9] Oppenheim, A.V., and Schafer, R.W. (1975). Digital Signal Processing: Prentice-Hall.

(Pages 110 through 113 of this textbook describe the overlap-add method used with the FDIC option in HIFRIC.)

<u>CORAVD</u>

[10] Basili,M. and Brady, A.G., (1978). "Low frequency Filtering and the Selection of Limits for Accelerogram Corrections": Proc. 6th European Conf. on Earthquake Engineering, Dubrovnik, Yugoslavia.

[11] Fletcher, J.B., Brady, A.G. and Hanks, T.C. (1980). "Strong-Motion Accelerograms of the Oroville, California, Aftershocks: Data Processing and the Aftershock of 0350 August 6, 1975": Bulletin of the Seismological Society of America, vol.70, No.1, pp.243-267.

[12] Hamming, R.W. (1977). Digital Filters: Prentice-Hall.

[13] Ormsby, J.F.A., (1961). "Design of Numerical Filters with Application to Missile Data Processing": Journal of the

Association for Computing Machinery, v.8, pp.440-446.

RSPECT

[14] Chopra, A.K. (1980). Dynamics of Structures; A Primer: Earthquake Engineering Research Institute, 2620 Telegraph Avenue, Berkeley, CA 94704. 126 pages.

[15] Newmark, N.M. and Hall, W.J. (1982). Earthquake Spectra and Design: Earthquake Engineering Research Institute, 2620 Telegraph Avenue, Berkeley, CA 94704. 103 pages.

[16] Nigam, N.C. and Jennings, P.C. (1969). "Calculation of Response Spectra from Strong-Motion Earthquake Records": Bulletin of the Seismological Society of America, vol. 59, pp. 909-922.

PHASE5

[17] Perez, V., (1973). "Velocity Response Envelope Spectrum as a Function of Time for the Pacoima Dam, San Fernando Earthquake, February 9,1971": Bulletin of the Seismological Society of America, vol. 63, pp.299-313.

PHASE6

[18] Perez, V. (1980). "Spectra of Amplitudes Sustained for a Given Number of Cycles: An Interpretation of Response Duration for Strong-Motion Earthquake Records": Bulletin of the Seismological Society of America, vol. 70, pp.1943-1954.

PHASE7

[19] Perez, Virgilio and Brady, A.G., (1984). "Reversing Cyclic Demands on Structural Ductility During Earthquakes" in "Proceedings of Applied Technology Council Seminar on Earthquake Ground Motion and Building Damage Potential; San Francisco": in press.