SANDIA REPORT

SAND98–0628 • UC–706
Unlimited Release
Printed March 1998

GRAFLAB 2.3 for UNIX
A MATLAB Database, Plotting, and Analysis Tool

User's Guide

William N. Dunn

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

SF2900Q(8-81)

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A04
Microfiche copy: A01

DISCLAIMER

Portions of this document may be illegible electronic image products. Images are produced from the best available original document.

SAND98-0628 **Distribution**
Unlimited Release **Category UC-706**
Printed March 1998

GRAFLAB 2.3 for UNIX
A MATLAB Database, Plotting, and Analysis Tool
User's Guide

William N. Dunn
Mechanical & Thermal Environments Department
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-0415

Abstract

This report is a user's manual for GRAFLAB, which is a new database, analysis, and plotting package that has been written entirely in the MATLAB programming language. GRAFLAB is currently used for data reduction, analysis, and archival. GRAFLAB currently runs on UNIX platforms supported by MATLAB. GRAFLAB was written to replace GRAFAID, which is a FORTRAN database, analysis, and plotting package that runs on VAX/VMS.

Contents

Illustrations

Tables

GRAFLAB 2.3 for UNIX
A MATLAB Database, Plotting and Analysis Tool
User's Guide

1.0 Introduction

GRAFLAB is a database, analysis, and plotting package written entirely in MATLAB, designed to replace the outdated GRAFAID code.[1] GRAFLAB is a superset of commands that works on top of the MATLAB matrix manipulation software. The user of GRAFLAB has the option of using either MATLAB routines or the added GRAFLAB routines to manipulate data.

The GRAFLAB routines are stored in subdirectories in the MATLAB path, and the GRAFLAB database is located under the current directory in a subdirectory called `alldata`. The database, analysis, and plotting routines are all located in several common directories that exist in the MATLAB path statement.

This report is a condensed tutorial of those commands that perform 80% of the GRAFLAB tasks. To get more extensive help concerning a specific `command.m` file, simply type in

```
help command
```

All GRAFLAB commands that are typed by the user, or output by the computer, are displayed in this manual in the Courier font as follows: `command(input);`.

1.1 Prepare to Use GRAFLAB

First, to start a MATLAB session that runs GRAFLAB 2.3, the startup.m file, which can be obtained from the code sponsor, must be placed in your `/home/username/matlab` directory. A sample `startup.m` file is shown in Appendix A.

Next, open the directory in which you want to run GRAFLAB and create a new subdirectory called `alldata`. It is in this subdirectory that the GRAFLAB database will store the curves and the information related to them.

Finally, type `MATLAB` at the UNIX prompt. On a UNIX platform, if the following output is displayed, the database is open.

```
*****************************************

     Welcome to GRAFLAB version 2.3U
               for UNIX

*****************************************

>>
```

However, if the following warning is displayed, the `./alldata` directory may have been entered improperly.

---

[1] Adams, C. R., "GRAFAID Code User Manual. Version 2.0," SAND84-1726 Unlimited Release UC-32, September 1985.

```
./alldata/ not found
```

At the MATLAB prompt, type

```
>>!mkdir ./alldata
```

1.2 Update the Database

If you start up GRAFLAB in a directory that contains an old database (GRAFLAB 1.0 or GRAFLAB 2.0), the `startup.m` file for GRAFLAB 2.3 will call `gl2update` to update the `Header.mat` file for each curve to the current GRAFLAB format. This is an irreversible process that will cause you problems if you try to go back to running an older version of GRAFLAB on this new database format. The most recent versions of GRAFLAB (2.1, 2.2 and 2.3) all use the same database format.

1.2a Update GRAFLAB database from version 1.0 to 2.3

When a version 1.0 database is encountered, the user is given a choice to update from version 1.0 to version 2.3 of the database and is warned that the process is irreversible.

```
***** WARNING:  GRAFLAB 1.0 DATABASE ENCOUNTERED *********

Updating the database is irreversible!!
Do you wish to update the database to version 2.3a ?
1. Yes
2. No
Enter 1 or 2
>1

Updating ./alldata/PPCYTC1/Header.mat
Updating ./alldata/PPCZTC1/Header.mat
Updating ./alldata/PPGZTC1/Header.mat
Updating ./alldata/PRDXTC1/Header.mat
Updating ./alldata/PRDYTC1/Header.mat
Updating ./alldata/PRDZTC1/Header.mat
Updating ./alldata/PTSXTC1/Header.mat
Updating ./alldata/PTSYTC1/Header.mat
Updating ./alldata/PTSZTC1/Header.mat
Entire GRAFLAB database updated to GRAFLAB 2.3a
```

1.2b Update GRAFLAB database from version 2.0 to 2.3

When a version 2.0 database is encountered, the user is given no choice, and the update is automatic (and minor). The curvename is removed from the `Header.mat`, and the curve legend is updated to include only the legend, not the curvename.

```
**** Minor header update necessary for version 2.3a
Updating ./alldata/ah25_10/Header.mat
```

```
Updating  ./alldata/dh25_1/Header.mat
Updating  ./alldata/sh25_1/Header.mat
Updating  ./alldata/shghav25/Header.mat
Updating  ./alldata/vh25_1/Header.mat
Updating  ./alldata/y/Header.mat
Entire GRAFLAB database updated to GRAFLAB 2.3
```

1.3 Improvements in GRAFLAB Version 2.3

GRAFLAB version 2.3 retains the functionality of previous versions, with many added enhancements. Your m-files should run as they did previously with only minor modifications.

The major differences between version 2.3 and 1.0 follow:

1. GRAFLAB 2.3 uses only MATLAB; that is, no MEX files (C code).
2. GRAFLAB 2.3 uses no master list of the database files. The database is now simply those subdirectories that appear under `./alldata`. Each curvename is the subdirectory name and each subdirectory must contain a `Header.mat`, which contains header information, and a `Data.mat`, which contains the data.
3. GRAFLAB 2.3 now allows 31 character curvenames.
4. The complexity of the database has been reduced:
   a. While GRAFLAB 1.0 used 140 m file (many of which were written in C, version 2.3 uses less than 40, all of which are written in MATLAB.
   b. Several GRAFLAB 1.0 routines were rewritten for speed and simplicity.
5. The utility function, `showd`, now returns an output argument (`nargout`).
6. The utility function, `gcopy`, enables copying of curvename subdirectories from remote databases.
7. A faster, more powerful `gdir` was developed for parsing curvenames.
8. Plot legends (and grids) now work in `bplot` using new globals.
9. Code is more easily portable to other platforms.
10. Plotting of complex numbers is enabled.

2.0 Basic GRAFLAB Usage

This section provides a basic tutorial on the use of GRAFLAB. However, the user is encouraged to become familiar with the basic MATLAB syntax and command structure before attempting to use GRAFLAB. Then, having mastered the basics of MATLAB, GRAFLAB will seem much more intuitive.

GRAFLAB makes extensive use of the MATLAB concepts of global variables, a local workspace, and the ability to save and load data from binary `.mat` files stored on disk. Therefore, it is important for the user to understand that while most analyses will be conducted exclusively in the MATLAB local workspace, GRAFLAB stores numerous parameters needed to define the user environment (Section 4.0) as global variables. In addition, when the user retrieves curves from GRAFLAB's `alldata` database (using the `setd('act')` command), these curves are maintained within GRFALAB as global variables.

2.1 Create and save a curve from within GRAFLAB

To create and save a GRAFLAB curve, first assign values to a MATLAB variable, zzz,

```
zzz=[1 2 ; 3 4 ; 5 6]
```

Define the following plot parameters: plot title (pt), the y label (pyl), the x label (pxl), the plot extremes (pe), and the axis type (at), using the GRAFLAB command `setd`. GRAFLAB assigns default values for all the plot parameters except the plot tile. These can be checked using the `showd` command.

```
setd('pt','A sample plot/Includes 5 lines of info/line
3/line4/line5')
setd('pyl','Y axis description here')
setd('pxl','X axis description here')
setd('pe','0/10/0/10')
setd('at','nolog')
```

Since GRAFLAB curves are stored as global variables, the user must declare a new curve to be a global variable before it can be saved.

```
global zzz
savecurf('zzz',0,6)
```

The curve zzz is now a part of your GRAFLAB database, containing both a header and data. Use `gdir` to view the database contents.

2.2 Load external files and save them into GRAFLAB

ASCII files can be loaded easily into GRAFLAB. If the files are columns of comma or tab delimited numbers in ASCII format, MATLAB has its own load command. First, load in the data. For example, an ASCII data set (CHAN1.DAT) with 2 columns (column 1 is the time, and column 2 is the acceleration in g's) will be loaded.

```
load CHAN1.DAT -ascii;
```

Then, when the data is loaded, perform a

```
whos
```

The variable CHAN1 and its dimensions (127,998 rows and 2 columns, in this case) will display.

Name	Size	Elements	Bytes	Density	Complex
CHAN1	127998 by 2	255996	2047968	Full	No

Now, plot column 2 versus column 1.

```
plot(CHAN1(:,1),CHAN1(:,2))
```

The data can be truncated, taking the first 100,000 pairs.

```
CHAN1=CHAN1(1:100000,:);
```

> **Note.** Do not omit the semicolon (;) that is located at the end of the path, or the data (100,000 pairs of points!) will scroll to the screen.

Now, save the data in CHAN1 to curve name "chan1," load and save CHAN2.

```
chan1=CHAN1;
global chan1
setd('pt','Channel 1')
setd('pyl','G''s')
setd('pxl','time (sec)')
setd('pe','0/.5/-2500/2500')
setd('at','nolog')
savecurf('chan1',0,6)
load CHAN2.DAT -ascii;
chan2=CHAN2;
global chan2
setd('pt','CHANNEL 2')
savecurf('chan2',0,6)
```

Now there are three curves in the database. Look at the names using gdir

```
gdir
 1. chan1
 2. chan2
 3. zzz
```

Now we can use

`showd('ch','chan1')`	To show the chan1 header values.
`setd('act','chan*')`	To put the curves into the active set.
`setd('pld','on')`	To display the legends of the two curves.
`setd('pg','on')`	To set the grid on.
`setd('plco','red/blue')`	To set the colors.
`bplot`	To plot the curves.

The database is saved to disk in `./alldata` as subdirectories. The utility `gdir` is the key to accessing the database. Perform a `help gdir`, and carefully read the help in this important utility.

2.3 Load a Neutral File into GRAFLAB

Neutral files are formatted ASCII text data files. To load a neutral file into GRAFLAB, use the following command:

```
ninput('./directory/filename.neu',overwrite,grafaidflag)
```

where overwrite = 1 if you want to overwrite existing curvenames in your current database with the curves read from `filename.neu`.

The `grafaidflag` identifies when a neutral file was written by GRAFAID (as compared to one written by GRAFLAB or some other source). This is important to know because, for some reason, GRAFAID neutral files omit the 'e' when writing out data in exponential format (for example 2.0e-3 is written 2.0-3). Unfortunately, GRAFLAB does not interpret this properly. The `grafaidflag` invokes a routine that accounts for the missing 'e' when loading in the data.

2.4 Write a Neutral File from GRAFLAB

To write a Neutral file from GRAFLAB, use the following command:

```
noutput('./directory/filename.neu','curvename')
```

GRAFLAB will prompt you to select the option to dump the curves that match gdir('curvename') from the database, and then to enter a neutral file title.

2.5 Generic ASCII File (with Headers/Text) Input

Use the GRAFLAB `gfile` utility to import ASCII files that contain both data and headers in formats other than neutral files.

2.6 GRAFLAB Curve Input

Use the GRAFLAB `gcopy` utility to copy GRAFLAB curves from one GRAFLAB database to another.

2.7 GRAFLAB Utilities

GRAFLAB utilities perform useful transforms on the input arguments to generate output. For example, the utility `gpsd.m` generates a power (or acceleration) spectral density, pxx, from a user specified acceleration history, x. For more information, type `help gpsd`.

syntax:

```
pxx=gpsd(x,blocksize,overlap,timerange,window)
```

The variable `x` is the input acceleration history. As is typical of most GRAFLAB utilities, if 'x' is missing, or null, `gpsd` will use the first curve in the active set. Similarly, any other argument will either have a designated default value or will require the user to provide values before the analysis can proceed. Arguments can be either numeric or text as dictated by the utility in question. Empty brackets `[]`, denote that the user wishes to use the designated GRAFLAB default value.

An example input is shown below:

```
[yenv]=gpsd([],1024,50,[],2);
```

3.0 The GRAFLAB Database

3.1 Database Design

The database design is simple yet functional. All of the curves are stored as subdirectories in the alldata subdirectory. The name of each subdirectory is the name of the curve. The header information, the data, and additional auxiliary information for a given curve is located under that subdirectory. Data is accessed by setting it active using the `setd('act','curvename')` command.

3.2 Database Structure

The GRAFLAB 2.3 database is shown in Figure 1. Each curve has a curvename, which is merely a subdirectory under the `./alldata` directory. The curvename subdirectory contains two or three MATLAB binary `*.mat` files: `Header.mat` and `Data.mat`, and optionally, an `Aux.mat` file. `Header.mat` contains 16 variables, which define primarily the plot parameters of curvename. The curvename length (number of characters) is a system dependent value equal to the variable CL. This length is a minimum of 39 characters. `Data.mat` contains the data in a variable that has the corresponding curvename. `Aux.mat`, although not fully implemented, contains auxiliary data for special uses.

`showd('ch','curve1')`	Displays the relevant `Header.mat` variables for `curve1`.
`showd('cd','curve1')`	Displays the data stored in `Data.mat` for `curve1`.

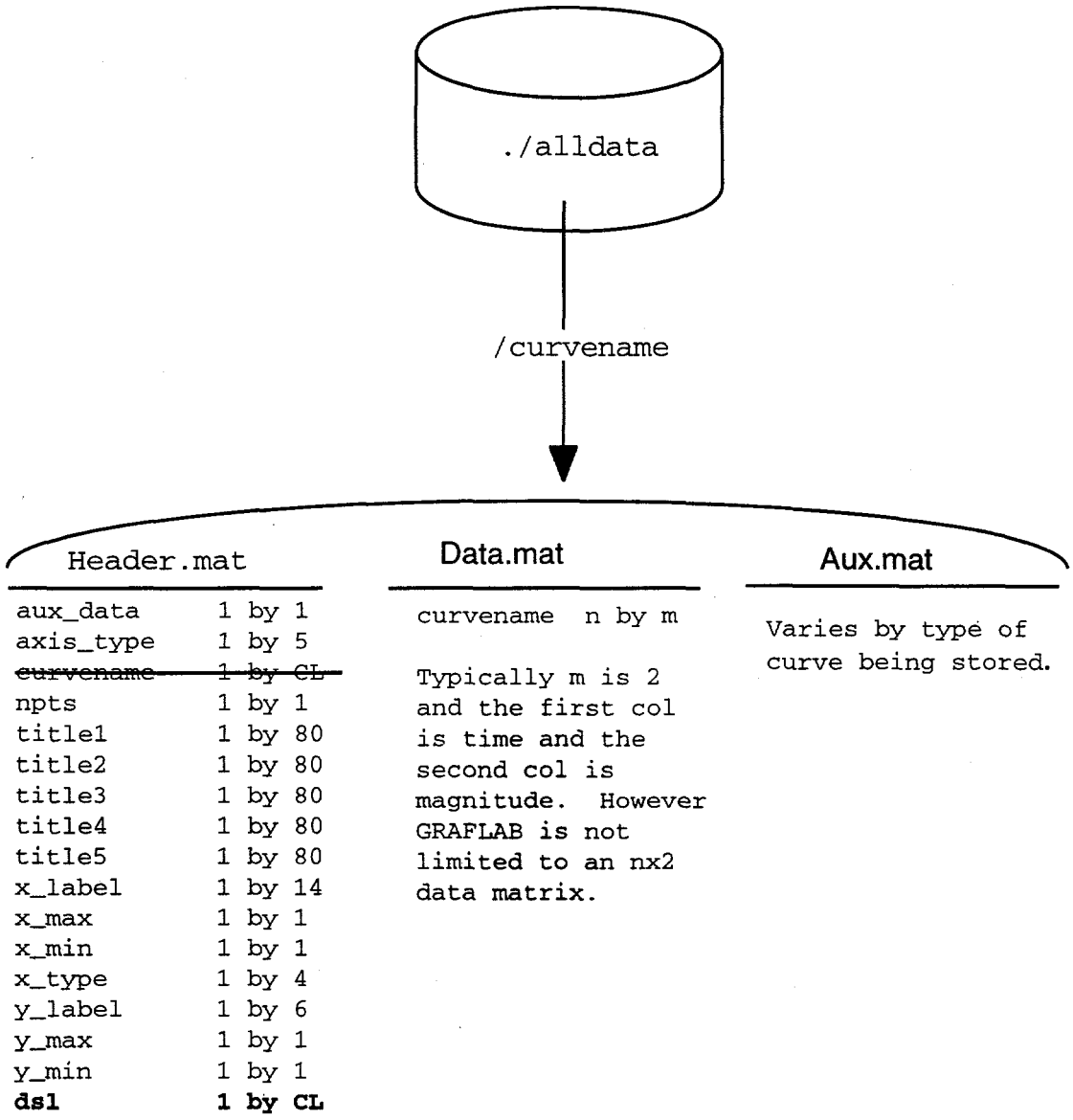


Figure 1. GRAFLAB Database Structure

3.3 Database m-files

The GRAFLAB database m-files are located in the directory:

/disk1/graflab23

These files are informally divided into two categories: user m-files and programmer m-files. For a brief description of each of these m-files, refer to sections 5.3.1 (user m-files) and 5.3.2 (programmer m-files).

3.4 CURVPARM – The System Dependent Function

The function `curvparm.m` defines the globals that control the database filenames, the system dependent delimiters, and max curve name length, etc. These globals can be accessed using the `curvparm` command. For example,

```
curvparm('DIR_PATH') - returns './alldata/' for UNIX.
```

SYSTEM Globals
(Defined in curvparm.m)

'NL'	System defined Newline Character.
'TRUE'	= 1
'FALSE'	= 0
'DIR_PATH'	Default Curve Directory Path.
'DIR_SEP'	System defined Directory Separator.
'NAME_MAX'	Maximum File Name Length.
'PATH_MAX'	Maximum Path Name Length.
'MACHINE_ID'	System Machine Identification.
'DIR_NAME'	Default Curve Names File.
'NEU_NAME'	Default Neutral Names File.
'HEAD_NAME'	Default Header File Name.
'AUX_NAME'	Default Auxiliary File Name.
'DATA_NAME'	Default Data File Name.
'SATADEF_NAME'	Default Satadef File Name.
'CNAME_LENGTH'	Curve Name Maximum Length.
'ENTRY_LENGTH'	Maximum Entry String Length.
'START_ENTRIES'	Initial # of Curve Entries in Curve File.
'EXT_ENTRIES'	Extend Curve Entry Increment.
'FILE_ID'	Standard File ID & Version.
'HAUX'	Haversine Auxiliary File Type.
'SAUX'	Decayed Sine Auxiliary File Type.

4.0 Plotting & Curve Manipulation

Plotting and curve manipulation parameters are controlled by the `setd` and `showd` commands, which modify the GRAFLAB globals. Because most of the GL globals deal with plotting, globals are discussed in this section.

4.1 Useful SETD and SHOWD commands

The `setd` and `showd` commands are used to assign values to global variables and to echo those values, respectively. SETD and SHOWD commands fall into two main categories: active set environment, and curve environment.

The first category includes those commands associated with the environment for the active (or plot) set. These commands are generally intended to define the active set and to tailor the look of the active set plot (`bplot`). These commands have the form `setd('p*',option)`, where the `'*'` represents one or two additional characters which, along with the `'p'`, form the acronym for the desired command.

The second category includes those commands associated with the environment for specific curves. These commands have the form `setd('c*',curvename,option)`, where the `'*'` represents one or two additional characters, which along with the `'c'`, form the acronym for the desired command. Many curve set commands have analogous plot set commands.

Some of the most common setd and showd commands, followed by a comprehensive listing, are shown below. Figures 2 and 3 summarize many of the plotting commands in graphical form.

SETD – Defines the GRAFLAB working environment. Screen output is often provided as an echo of the parameter that has just been set. Some useful commands follow:

`setd('act',curvename)` – Used to put a curve in the active set. setd('act','PA*') will put all the curves matching the pattern into the active set. setd('act','PAGPSX','add') will add the curvename to the existing active set.

`setd('af',option)` – Sets the axis format where option is as follows:

TICAXES	Produces tics only on X and Y axes.
TDASH	Produces tics on all sides at minor tic divisions and dashed grids at the major tic divisions.
TDOT	Produces tics on all sides at minor tic divisions and dotted grids at the major tic divisions.
TLINE	(DEFAULT) produces tics on all sides at minor tic divisions and solid grids at the major tic divisions.
GRID	Produces solid grid lines.
DASH	Produces dashed grid lines.
PLAIN	Produces no tics or grids.

`setd('at',option)` – Sets the axis type. This can also be set permanently for individual curves using

`setd('cat',curvename,option)` where option is

NOLOG	Linear/Linear Axes
XLOG	X Log/Y Linear Axes
YLOG	X Linear/Y Log Axes
XYLOG	Log/Log Axes
POLAR	Polar Axes

`setd('cl',curvename,newlegendtext)` – Sets the curve legend.

`setd('cn',curvename1,curvename2)` – Renames curve.

`setd('ct',curvename,titletext)` – Sets the curve title.

`setd('cyl',curvename,text)` – Sets the curve y label.

`setd('cxl',curvename,text)` – Sets the curve x label.

`setd('pe',text)` – Sets the plot extremes, which can be a text string `'xmin/xmax/ymin/ymax'`, or they can be a vector.

```
[xmin xmax ymin ymax]
```

`setd('pld',text)` – Sets the plot legend display.

Text can be 'on', 'on1', or 'off'

`setd('pld','on')` – Displays the legend if numcurves active > 1.

`setd('pld','on1')` – Displays the legend if numcurves active >= 1.

`setd('pld','off')` – Does not display the legend.

`setd('pll',option)` – Sets the plot legend location in percent of the screen. The variable option can be a string '0.3/0.6', or a vector [0.3 0.6]. The variable option can also be 'AUTO', in which case MATLAB will reliably auto locate the legend box to avoid covering the data.

Caution: This procedure may require 15 to 20 seconds on an HP735 for MATLAB to calculate a good position if you are plotting several hundred thousand points.

`setd('plco',text)` – Sets the plot line colors. The variable text can be `'YELLOW/MAGENTA/CYAN/RED/GREEN/BLUE/WHITE/BLUE'`.

`setd('plto',text)` - This sets the sequence of the dashes used to plot each curve. Text can be `'solid/dashed/dashdot/dotted/none'`.

```
setd('plto','solid/dashed/dashdot/dotted/none')
```

`setd('pls',text)` – This command does not appear to work, however, it is supposed to allow you change the symbols to the Lucky Charms marshmallow shapes. The allowable symbols are `' .ox+*'`

`setd('plt',text)` – This allows you to set the permutations of the plot lines:

```
'DASH'/'DSYM'/'LINE'/'LSYM'/'SYMB'/'NSYM'

/':NUL'/':SYM'/'D:NUL'/'D:SYM'
```

`setd('plw',text or num)` – Sets the line width. Seems to have a "step" effect for bitmap and screen display, but has a nice gradual effect for vector rendering formats such as POSTSCRIPT. This feature is nice for viewgraphs.

`setd('pn',text)` – Sets the plot name over the top of the figure window.

`setd('pp',text)` - Sets the figure position.

`setd('pt',text)` – Sets the plot title.

`setd('pxl',text)` – Sets the plot x label.

`setd('pyl',text)` – Sets the plot y label.

`setd('qa',text)` – Sets the qa display ('on' or 'off').

`setd('qad',text)` – Sets the qad display ('on' or 'off').

SHOWD – Displays current GRAFLAB working environment. The `showd` command provides output if an output argument is included. Data input that can be set using `setd`, can be shown using `showd`. Some useful commands follow:

SHOWD permits the use of an output argument, nargout, having the following form:

```
nargout = showd('*',option)
```

In each case, the output argument provides useful data to the user for the particular SHOWD command being invoked. For example:

```
x=showd('cd',curve1);
```

populates 'x' with a copy of the curve data comtained in curve1).

`showd('act')` – Shows the curvenames in the active set.

`showd('cd',curvename,range)` - Echoes to the screen the curve data contained in the string curvename over the span of points range. If you have an acceleration history over curve called 'ACC' and you want to put the data from time = 0.2 to 3.2 seconds in a variable named y, the following command would do it:

```
y=showd('cd','ACC','0.2/3.2')
```

`showd('ct',curvename)` – Echoes the curve title. For wildcard viewing of multiple curve titles using wildcards use
`showd('ct',gdir('PTSZ*'))`

`showd('ch',curvename)` – Echoes the curve header for curvename. This is useful if you need more than one piece of information about a curve.

`showd('users')` – Shows the users, their process id, and the CPU time. Also can use `who_mat`. This command is useful because MATLAB occasionally fails to relinquish licenses after a process is inadvertently killed.

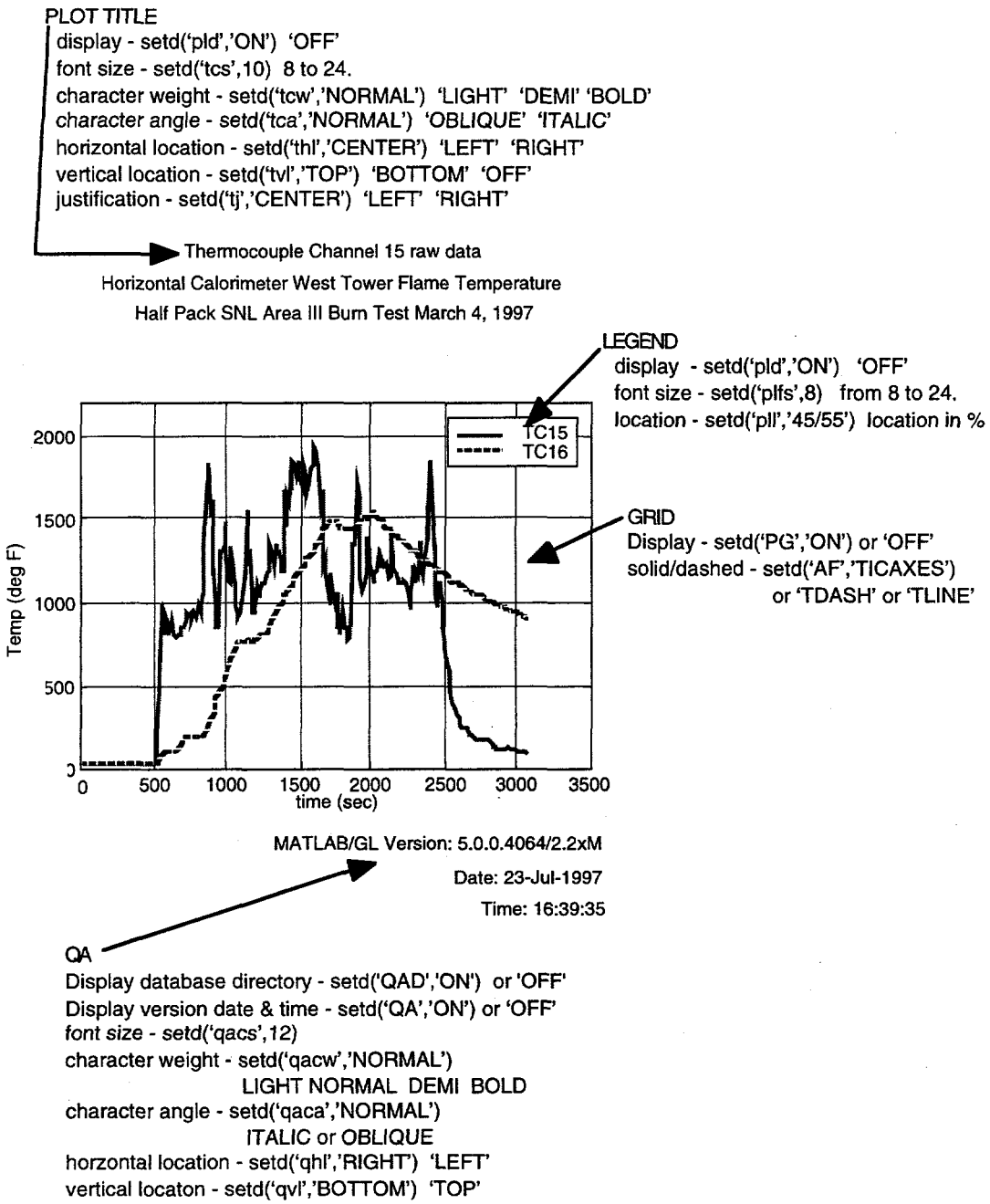
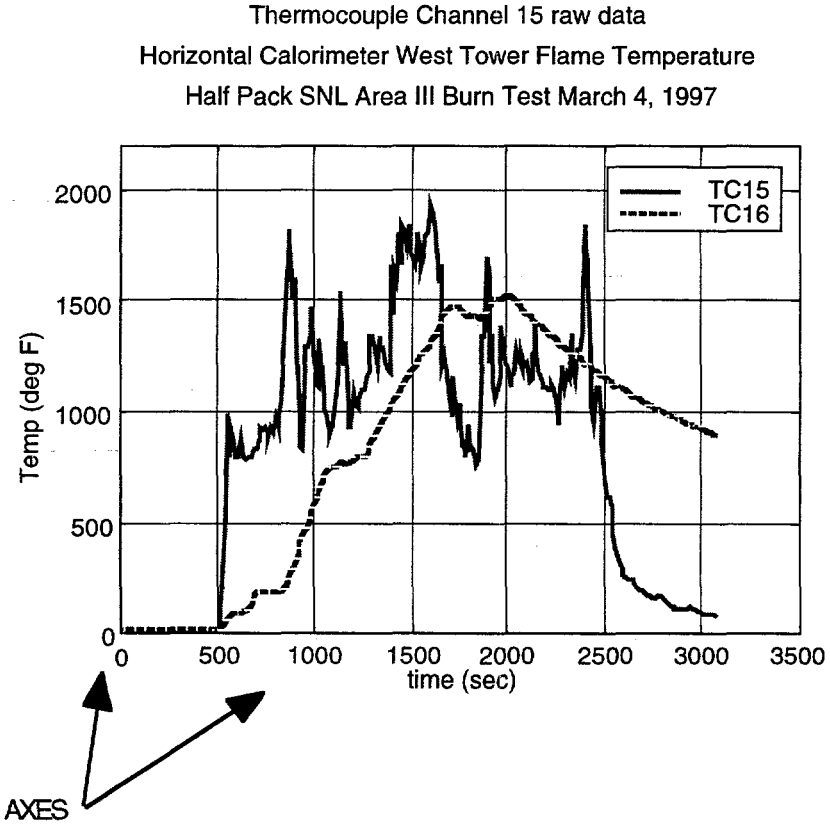


Figure 2. Plot Commands by Location

PLOT CONTROLS

Background color - setd('BGC','BLACK') 'WHITE'
Plot extremes - setd('pe','0/10/1e-5/0.1')
line colors - setd('plco','RED/BLUE')
'YELLOW/MAGENTA/CYAN/RED/GREEN/BLUE/WHITE/BLACK'
line style - setd('plto','SOLID/DASHED') 'DOTTED/DASHDOT'
symbols - setd('pls','.') 'o', 'x', '+' or '*'
line type - setd('plt','LSYM')
DASH/DSYM/LINE/LSYM/SYMB/NSYM/:NUL/:SYM/D:NUL/D:SYM
line width - setd('lw',1.5)
plot name - setd('pn','Plot name')
plot location - setd('pp',[49 99 20 70])
x label text - setd('pxl','Plot x label')
y label text - setd('pyl','Plot y label')



AXES

Label angle - setd('ALCA','NORMAL') or 'ITALIC' or 'OBLIQUE'
Font size - setd('ALCS',12) - size in points (1/72 inch).
Char weight - setd('ALCW','NORMAL') - 'LIGHT' 'DEMI' or 'BOLD'
scale - setd('AS','AUTO') - 'OWN' or 'FULL'
type - setd('AT','NOLOG') - 'XYLOG' 'YLOG' 'XLOG' or 'POLAR'
xlabel display - setd('AXL', 'XBOTH') 'XNUMERIC' 'ALPHA' or 'NOXLABELS'
ylabel display - setd('AXL', 'XBOTH') 'XNUMERIC' 'ALPHA' or 'NOXLABELS'

Figure 3. Plot Commands by Location

Note that showd returns an nargout value that can be used.

To get the data stored in curve1 and store it in y,

```
y=showd('cd','curve1');
```

To set the curve x label of curve2 to the same as curve1, type

```
setd('cxl','curve2',showd('cxl','curve1'));
```

There is a considerable amount of data processing power using gdir and a MATLAB `for` loop. To set the curve x label of all the curves named Curve* to the same as curve1,

```
a=gdir('Curve*');

for  i=1:size(a,1)

     setd('cxl',clearaws(a(i,:)),showd('cxl','curve1'));

     end
```

4.2 Comprehensive SETD and SHOWD command listing

4.2.1 Plotting Commands

Command	Abbrev.	Description / Options
ACTIVE SET	(ACT)	Puts a list of curves into the active set and puts their names into the global activecurves. Use `setdloc` to bring the data into the local workspace named according to the curvename.
AXIS FORMAT	(AF)	`ticaxes` – dotted grid lines `tdash` – dashed grid lines `tline` – solid grid lines
AXIS LABEL ANGLE	(ALCA)	Normal – Italic – Oblique –
AXIS LABEL SIZE	(ALCS)	Size in Points (1/72 inch)
AXIS LABEL WEIGHT	(ALCW)	Light – Normal – Demi – Bold –
AXIS SCALE	(AS)	own – auto – full –

AXIS TYPE	(AT)	NOLOG – XYLOG – YLOG – XLOG – POLAR –
AXIS X LABEL TYPE	(AXL)	XBOTH – Displays both alpha & numeric. XNUMERIC – Displays only numbers. ALPHA – Displays only text. NOXLABELS – Displays no labels.
AXIS Y LABEL TYPE	(AYL)	XBOTH – Displays both alpha & numeric. XNUMERIC – Displays only numbers. ALPHA – Displays only text. NOXLABELS – Displays no labels.
BACKGROUND COLOR	(BGC)	Black White
FONT TYPE	(FT)	Helvetica Courier Times Symbol Avant Garde Bookman Helvetica Narrow New Century Schoolbook Palatino Zapf Chancery Zapf Dingbats
PLOT EXTREMES	(PE)	Sets the 'xmin/xmax/ymin/ymax' for plotting.
PLOT GRID	(PG)	Sets the grid display on or off. ON OFF
PLOT LEGEND DISPLAY	(PLD)	Sets the display of the Plot Legend on or off. ON - Will display if more than one curve. OFF - Will not display legend.
PLOT LEGEND FONT SIZE	(PLFS)	Sets the plot legend font size. Acceptable values are from 8 to 24
PLOT LEGEND LOCATION	(PLL)	Sets the location of the plot legend in percent of the plot screen. [.6 .3] – 60% along x, 30% along y. '.6/.3' – 60% along x, 30% along y.
PLOT LINE COLORS example: setd('plco','red/yellow')	(PLCO)	y = 'YELLOW '; m = 'MAGENTA'; c = 'CYAN '; r = 'RED '; g = 'GREEN '; b = 'BLUE '; w = 'WHITE '; k = 'BLACK ';

PLOT LINE ORDER example: setd('plto','solid/solid')	(PLTO)	Determines whether the plots will be dashed, solid, dot-dash, etc. none = ' '; solid = '- '; dashed = '--'; dotted = ': '; dashdot = '-.';
PLOT LINE SYMBOLS	(PLS)	'.' 'o' 'x' '+' '*'
PLOT LINE TYPE	(PLT)	DASH– Dashed no symbols. DSYM– Dashed with symbols. LINE– Solid lines with no symbols. LSYM– Solid lines with symbols. SYMB– Same as LSYM. NSYM– Symbols only. :NUL– Dotted plot lines nosymbols. :SYM– Dotted plot lines with symbols. D:NUL– Dashdot Plot Lines no symbols. D:SYM– Dashdot Plot lines with symbols.
PLOT LINE WIDTH	(PLW)	Sets the line width of the curves in Numeric Points (1/72 inch).
PLOT NAME	(PN)	Sets the name of the Figure Window.
PLOT POSITION (FIGURE)	(PP)	Sets the location of the Figure Window.
PLOT TITLE	(PT)	Sets the plot title. 'LINE1/LINE2/LINE3/LINE4/LINE5'
PLOT X LABEL	(PXL)	Sets the plot x axis label.
PLOT Y LABEL	(PYL)	Sets the plot y axis label.
TITLE CHAR ANGLE	(TCA)	Normal Italic Oblique
TITLE CHAR SIZE	(TCS)	Size in Points (1/72 inch)
TITLE CHAR WEIGHT	(TCW)	Light Normal Demi Bold
TITLE HORIZONTAL LOCATION	(THL)	Left Center Right
TITLE JUSTIFICATION	(TJ)	Left Center Right
TITLE VERTICAL LOCATION	(TVL)	Sets the plot title vertical location.

4.2.2 Curve Header Commands

Command	Abbrev.	Description / Options
CURVE AXIS TYPE	(CAT)	NOLOG – XYLOG – YLOG – XLOG – POLAR –
CURVE DATA	(CD)	Echoes the curve data to the screen or to `nargout`.
CURVE EXTREMES	(CE)	Sets the 'xmin/xmax/ymin/ymax' for plotting.
CURVE LEGEND	(CL)	Sets the curve legend to a string to be used in plotting.
CURVE NAME	(CN)	Renames the curve. `setd('cn','curve1','curve2')`
CURVE TITLE	(CT)	Sets the curve title `setd('ct','curvename','LINE1/LINE2/LINE3/LINE4/LINE5')`
CURVE X LABEL	(CXL)	Sets the x label of the curve.
CURVE X TYPE	(CXT)	X axis data type. MONO - Monotonically increasing data. NONM - Non monotonic data.
CURVE Y LABEL	(CYL)	Sets the y label of the curve.

4.2.3 QA Commands

Command	Abbrev.	Description / Options
QA CHAR ANGLE	(QACA)	Normal Italic Oblique
QA CHAR SIZE	(QACS)	Size in Points (1/72 inch)
QA CHAR WEIGHT	(QACW)	Light Normal Demi Bold
QA DATABASE	(QAD)	Toggles the display of the database title. Currently does not work in GRAFLAB 2.3. ON OFF
QA DISPLAY	(QA)	Toggles the display of the QA database. ON OFF
QA HORIZONTAL LOCATION	(QHL)	Left Right

4.3 GRAFLAB Globals

Shown below is the GRAFLAB 2.3 list of globals, and a description of each one. Typically, the user will employ `setd` to set the globals and `showd` to display their values. This section is included primarily as reference for programmers, or for those who want to know how GRAFLAB works in more detail.

WARNING: If a user performs a `clear globals` in MATLAB, then all of the globals shown below are cleared, and GRAFLAB will not function properly until MATLAB is restarted.

4.3.1 Useful Commands

`whos global`	To peek at all the hidden globals.
`shoglob('LINE_WIDTH')`	To echo the value of the global.
`global BACKGROUND_COLOR`	To make the globals accessible to the local space.

4.3.2 Plotting Globals

Use `whos global` to list plotting globals in MATLAB. To add a new global, see the programmer's section.

`'activecurves'`	A list of the active curves.
`'ACTIVE_FLAG'`	Active Set Plot.
`'AXIS_AREA'`	Screen Plot Area.
`'AXIS_FORMAT'`	Type of X Y Grid System.
`'AXIS_LABEL_ANGLE'`	Plot Label Character Angle.
`'AXIS_LABEL_SIZE'`	Plot Label Character Size.
`'AXIS_LABEL_WEIGHT'`	Plot Label Character Weight.
`'AXIS_SCALE'`	Type of Axes Scaling.
`'AXIS_TYPE'`	Plot Axis Type.
`'AXIS_X_LABEL'`	Plot X Labels Display.
`'AXIS_Y_LABEL'`	Plot Y Labels Display.
`'BACKGROUND_COLOR'`	Plot Background Color.
`'COMPLEX_DATA_TYPE'`	Flag for Real vs Imag, Mag, or Phase.
`'DECILOG'`	Decimate Log Status (ON/OFF).
`'DECIMATE'`	Decimate Bins/Factor.
`'DECITEST'`	Decimate Test Status (ON/OFF).

'DEVICE_GRID'	Plot Grid Status (ON/OFF).
'FONT_TYPE'	Plot Character Font Type.
'LINE_WIDTH'	Plot Line Width.
'PLOT_EXTREMES'	Plot Axes Limits.
'PLOT_LEGEND'	Legend Text.
'PLOT_LEGEND_BOX	Scale factor for Plot legend.
'PLOT_LEGEND_FLAG'	Legend Text Display ('ON' 'OFF').
'PLOT_LEGEND_FS'	Plot legend font size.
'PLOT_LEGEND_LOC'	Legend Location in %. [0.25 0.25].
'PLOT_LINE_COLORS'	Plot Colors.
'PLOT_LINE_ORDER'	Plot Line Types.
'PLOT_LINE_SYMBOLS'	Plot Symbols.
'PLOT_LINE_TYPE'	Plot Line Type.
'PLOT_NAME'	Plot (Figure) Name.
'PLOT_TITLE'	Plot Title Lines.
'PLOT_X_LABEL'	Plot X Label.
'PLOT_Y_LABEL'	Plot Y Label.
'QA_CHAR_ANGLE'	QA Character Display Angle.
'QA_CHAR_SIZE'	QA Character Display Size.
'QA_CHAR_WEIGHT'	QA Character Display Weight.
'QA_DATABASE'	QA DataBase Name Switch(ON/OFF).
'QA_DISPLAY'	QA Display Switch(ON/OFF).
'QA_HORZ_LOC'	QA Horizontal Location.
'QA_VERT_LOC'	QA Vertical Location
'TITLE_CHAR_ANGLE'	Title Line Character Angle.
'TITLE_CHAR_SIZE'	Title Line Character Size.
'TITLE_CHAR_WEIGHT'	Title Line Character Weight.
'TITLE_HORZ_LOC'	Horizontal Title Location.
'TITLE_JUST'	Title Justification.
'TITLE_VERT_LOC'	Vertical Title Location.

4.3.3 Calculation Globals

'AUX_TYPE'	Shock Spectra Type.
'CURVE_DEFAULT'	Curve Definition.
'DAMPING'	Damping (Decay Rate) Value.

'GRAVITY_CONSTANT'	Value to use for 'G'.
'INTERPOLATION_TYPE'	Interpolation Parameter.
'MATCH_CHARACTER'	Match Character.
'NUMBER_SAMPLES'	Number of Samples Value.
'QUERY'	Query flag.
'ROUND'	X/Y Rounding Places.
'SAMPLE_RATE'	Sample Rate.
'SS_TYPE'	Shock Spectra Parameters.
'TOLERANCE'	Tolerance.

4.3.4 Other Globals

'WILD_CHARACTER'	Wild Card Character.

5.0 GRAFLAB m-files

This section documents all the GRAFLAB m-files and many useful MATLAB m-files. GRAFLAB analysis functions that start with a "g" imply that they have been accepted as general "quality assured" m-files. Those that start with other letters (by convention, the first letter of the programmer's given name) are subject to modification and/or have not been accepted as a standard GRAFLAB utility.

5.1 General Notes on GRAFLAB m-files

1. GRAFLAB performs much of its housekeeping tasks in the MATLAB global workspace. However, manipulation of data must be done in the MATLAB local workspace.
2. GRAFLAB utilities are written for the most part as MATLAB functions and therefore, follow the corresponding Matlab conventions.
3. GRAFLAB uses the concept of an active set (just as with GRAFAID). Unless otherwise noted, GRAFLAB utilities will operate on either GRAFLAB curves in the active set or data in the local Matlab workspace.
4. Unless otherwise noted, GRAFLAB expects an nx2 matrix: the first column is the independent variable (usually assumed to be time in seconds or frequency in Hertz), the second column is the dependent variable (often assumed to be acceleration in G's for purposes of automatic curve labeling).
5. MATLAB works on row indices when working with the dependent variable (time). To permit the user to work with time directly, several utilities have been developed to make the conversion in the background.
6. A great strength of MATLAB is its ability to use `FOR` loops to perform repetitive tasks.
7. Curve data is edited using standard MATLAB matrix manipulation (refer to section 8, example 4 for a sample editing session).
8. GRAFLAB takes advantage of the Matlab HELP file syntax. Therefore, the user may display a help file for any specific m file by typing `help mfilename`. Similarly, the user may display a listing of all of the available m files within a GRAFLAB toolbox by typing `help toolboxname` (refer to Section 5.2 below for an explanation of available toolboxes).

5.2 Analysis m-files

GRAFLAB utilites are partitioned in a manner that is analogous to Matlab's toolboxes in order to provide the reader with a useful grouping of like routines. Figure 4 presents the preferred grouping of utilites. Only the utilities in the `gmechanical` and `gthermal` toolboxes will be discussed in this manual

The `gshaker` toolbox contains the m-files needed to synthesize shock pulses for use in shaker tersting. The `gmechanical` utility `gdsine` was written to act as a front end to one of the main sets of gshaker synthesis routines.

The `caputil` and `dosutil` toolboxes contain many useful routines not considered suitable for inclusion in GRAFLAB at this time and are therefore left under the control of a specific GRAFLAB user. The m files contained in these toolboxes are either considered to be work in progress (i.e., GRAFLAB utilites that have not been completely checked out) or Matlab m files that have been completely checked out but do not adhere to the GRAFLAB

syntax. The testcases toolbox contains m files that serve as quality assurance checks for the various utilities as well as useful examples of syntax and intended usage.

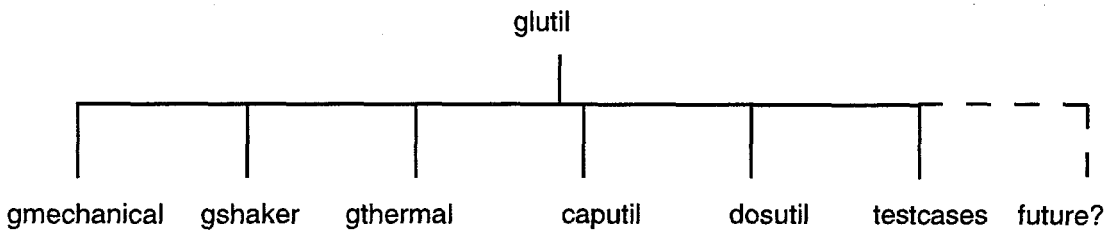


Figure 4. GRAFLAB Utility Directory Tree

Other toolboxes (both formal and user) will be added to GRAFLAB as the need arises.

In order to achieve the full functionality described in this manual, the GRAFLAB 2.3 STARTUP.M file must point to the following directories:

1. `disk1/glutil/gmechanical`
2. `disk1/glutil/gthermal`
3. `disk1/glutil/gshaker`
3. `disk1/glutil/testcases` (m-files to help quality assure utility output, and to serve as examples of intended usage).

Only the first two directories will be discussed in detail.

5.2.1 Mechanical Shock & Vibration (gmechanical)

`avdconv`	Routine to support avdplot.
`avdgrid`	Routine to support avdplot.
`avdplot`	Plots Shock Response Spectra on tri-coordinate paper
`dsd`	Computes the displacement spectral density (DSD) from the corresponding acceleration spectral density (ASD/PSD).
`gtempmom`	Computes the temporal moments for a time history.
`gbutterfilt`	Digital Butterworth lowpass filter.
`gclip`	Computes +/-n sigma envelopes based on a running rms for a time history and returns a time history that is the minimum of the original curve and the envelopes. The utility is useful for mitigating the effects of numerous data dropouts (see also gspike).

gconvolve	Computes an output time history from an input time history and frequency response function using the overlap and add method.
gcse	Computes the curve set data extremes (xmin, xmax, ymin, ymax) for the active set curves.
gdsine	Computes a decayed sine acceleration history, whose shock response spectra approximates a user defined reference shock response spectra. An auxiliary parameter file containing the associated input parameters is also generated.
gdsine_rpt	Generates a formatted report for a decayed sine parameter file (see gdsine).
genvelope	Produces a max or min envelope from a set of data curves,
genvelope_fatigue	Computes a PSD envelope and associated test duration that will produce the same amount of fatigue damage as a set of user defined PSDs and their associated test durations. The equivalencing procedure is based on a Minor's rule scaling law.
ghaversine	Computes the shock response spectra for a user defined haversine pulse.
gintegrate	Computes the integral of a time history using one of two techniques: a fourier transform technique or MATLAB cumsum.
gpsd	Computes the acceleration or power spectral density for an input acceleration history.
gpsd_bp	Computes the minimum number of break points needed to uniquely define a uniformly oversampled reference PSD assuming straight line on a log-log plot.
gpsd_rpt	Generates a formatted report containing the break points for a PSD. This file is intended primarily for use with straight line envelopes.
gpsdoct	Computes an octal bandwidth PSD from a linear bandwidth PSD.
grmsavd	Computes the acceleration, velocity, and displacement rms values for a PSD.
gsample	Produces a uniformly sampled matrix of points for a set of input curves.
gsmooth	Computes the smoothed version of a curve using a running average

gspike	Identifies the highest amplitude points in a time history and scales them using a user defined scale factor. This utility is designed to assist the user in identifying a finite set of data dropouts and noise spikes (see also gclip).
gsrs	Computes the shock response spectra for a time history.
gsrs_rpt	Generates a formatted report containing the break points for a shock response spectra. This file is intended primarily for use with straight line envelopes.
gtol_band	Computes the +/-N dB tolerance bands for a user defined reference curve.
gtranspose	Generates the two-sided (transposed or 0-2π) frequency response function from a one-sided (0-p) frequency response function.
gxspac	Computes the delta x value (and corresponding sample rate) and evaluates whether the input vector x is spaced evenly.
gxval2nrow	Identifies the row indices based on a set of user defined time (dependent variable) values. This utility is used to define a desired time range in terms of the row indices for use in Matlab.
vsd	Computes the velocity spectral density (VSD) from the corresponding acceleration spectral density (ASD/PSD).
gpsdbin	Computes a coarser linear bandwidth averaged version of a linear bandwidth PSD.
greconstruct	Generates a higher resolution curve from an input curve using reconstruction techniques
gwaterfall	Generates a waterfall plot (Matlab Specgram).
gwavsyn	A function to synthesize a transient composed of sums of WAVSYN wavelets that will match a specified shock response spectrum
rfft	Finds the fft of a real time history.
rfffti	Finds the inverse fft of a spectrum for a real time history.

5.2.2 Thermal Data Reduction Utilities (gthermal)

sodDFT	Thermal DR file to calc qi from DFT thermocouples.
sodqvts	Thermal DR file to combine q vs surf temp.
sodreadplt	Thermal DR file to read a SODDIT .plt file.
sodsaveall	Thermal DR file to save the input TC matrix.
sodwriteinp	Thermal DR file to write a SODDIT .inp file.

5.3 Database m-files

The "m-files," which are described below make up the GRAFLAB23 database structure. They are grouped by categories. They are found in the following directory on the Department 9735 HP735 server:

```
/disk1/graflab23
```

5.3.1 Database – User m-files

delcurf	Deletes a curve from the GRAFLAB 2.x database.
gcopy	Copies curve(s) from another GRAFLAB database to the current database.
gdir	Provides a selectable directory of the curves in the GRAFLAB database.
gfile	General file input with headers and various delimiters.
global	Useful for making GRAFLAB curves visible both to the local Matlab workspace and other functions.
gnargin	Checks the size and data type for any matlab variable. This m-file is intended to be used by programmers to verify that the user has passed a valid input argument to a GRAFLAB utility.
ninput	Inputs a neutral file into the GRAFLAB 2.x database.
noutput	Writes out a neutral for selected curves.
saveaux	Moves an auxiliary file (stored in MATLAB .mat format) from the user's present working directory (pwd) to the corresponding GRAFLAB database (./alldata/cname/Aux.mat).
savecurf	Saves header, data, and/or Aux data for a curve.
setd	Sets GRAFLAB globals, and curve attributes.

`setdloc`	Makes the active set curves visible to the Matlab local workspace.
`showd`	Shows the GRAFLAB globals, and curve attributes.

5.3.2 Database – GRAFLAB m-files for programmers

These files will never be called by the average user, but they are documented here for completeness.

`fixPLOTparm`	Fixes the plot parameters upon setactive.
`gl2update`	Updates the database or a curve to the current version.
`loaddata`	Loads the data from a curve.
`loadheader`	Loads the header from a curve.
`load_save.txt`	A text file describing the programmer's use of load & save.
`saveheader`	Saves the header of a curve.
`setactve`	Called by setd('act','...').
`setcvext`	Sets the curve extremes.
`setplext`	Sets the plot extremes.
`str2arry`	Parses an input string into a real array.
`str2mstr`	Parses an input string into a string matrix.
`sys`	Provides platform independent system calls, such as list files, `list` directories, copy, move, make directories, or delete directories.
`template`	A template used for writing new m files.

5.4 File I/O m-files

The "m-files," which are described below make up the GRAFLAB23 database structure. They are grouped by categories. They are found in the following directory on the Department 9735 HP735 server:

`/disk1/graflab23`

`gfile`	Loads ASCII files with headers and data.
`load`	Loads Matlab mat files or ASCII files (tab or space delimited).
`ninput`	Reads in Neutral files.
`noutput`	Writes out Neutral files.
`save`	Writes out Matlab mat files or ASCII files (space delimited).
`savecurf`	Saves header, data, and/or Aux data for a curve.
`readline`	Read in one line of an open file as text.
`readtext`	Puts the contents of a file into a text matrix.
`writetext`	Writes out a real or string matrix to a ASCII file.
`writeuff58`	Writes out a universal file format 58.

5.5 Plotting m-files

The "m-files," which are described below make up the GRAFLAB23 database structure. They are grouped by categories. They are found in the following directory on the Department 9735 HP735 server:

`/disk1/graflab23`

`bplot`	Plots GRAFLAB active set using GRAFLAB globals that define the plot. See Matlab plot command to plot local working variables.
`glegend`	Called from bplot to a curve legend string for each curve in the active set. Use setd('pld','on') to set bplot's legend display on.
`ggrid`	Creates a major grid pattern on a bplot. This was created as an alternative to the Matlab grid option (which plots both major and minor grids). Use setd('pg','on') to turn on grid display.
`plot2`	Plots an nx2 matrix, second column vs. first.

5.6 String Manipulation m-files

The "m-files," which are described below make up the GRAFLAB23 database structure. They are grouped by categories. They are found in the following directory on the Department 9735 HP735 server:

`abs(text)`	Echoes ASCII values for each character.
`blanks(num)`	Returns a string num spaces long.
`clearaws(string)`	Clears leading & trailing spaces of string (not internal spaces).
`clearws`	Clears leading white space (ASCII char 32) in string.
`deblank(string)`	Removes spaces at beginning.
`findstr('/',string)`	Returns a vector containing all locations of '/' in string.
`isempty(string)`	Returns 1 if string is [].
`~isempty(string)`	Returns 1 if string is not [].
`isletter(string)`	Returns 1 if is an alphabetic letter.
`isspace`	Returns 1 for space, newline, carriage return, tab, vert tab or formfeed.
`item(num,instr,delim)`	Parses a string by items and returns item n.
`readtext(filename,delim)`	Reads in a file as a text matrix.
`readtext(filename,delim)`	Reads in a file as a text matrix.
`setstr(textstring)`	Restores a string to text echo.
`str2mat`	Concatenates two strings (or text matrices).
`word(n,string)`	Parses a string by words and returns word n.
`wildcard`	Finds the location of a string in a text Matrix.
`writetext(filename,TextMat)`	Writes out a real or string matrix as a text file.

Example: Suppose you want to read from the text file named "input.dat" containing a filename, and the time range of the data, and use that time range to choose a portion of the data.

Text File: "input.dat"

```
TIMEFILE.DAT,10:05:05-15
```

The MATLAB syntax for this would be

```
theinput=readtext('input.dat',10);
% Note:  theinput now is 'TIMEFILE.DAT,10:05:05-15'
% The first item of theinput (by comma delim) is 'TIMEFILE.DAT'
datafile=item(1,theinput,',');
eval(['load ' datafile]) % load the data in from 'TIMEFILE.DAT'
% Get the first item (by period delim) which is 'TIMEFILE'
dataname=item(1,datafile,'.');
```

```
eval(['thedata= ' dataname ';']) % put the data into thedata
%
%  Now we need to truncate the data according to the time.
timerange=item(2,theinput,',');
% timerange is now '10:05:05-15'
secondrange=item(3,timerange,':');
% second range is now '05-15'
begintime=item(1,secondrange,'-');
endtime = item(2,secondrange,'-');
% begintime is now '05' and endtime is now '15'
% NOW TRUNCATE THE DATA USING GXVAL2NROW.
thedata=gxval2nrow(thedata,[ begintime,endtime ]');
```

5.7 GLOBAL m-files

The "m-files," which are described below make up the GRAFLAB23 database structure. They are grouped by categories. They are found in the following directory on the Department 9735 HP735 server:

`/disk1/graflab23`

`actparm`	Returns the requested default global variable value.
`setdglob`	Initializes the global variables (call from startup.m).
`setdloc`	Makes curves in active set global & visible to local space.
`curvparm`	Returns the requested system parameter.
`shoglob`	Returns the current value of the requested global variable.

5.8 Other m-files

The "m-files," which are described below make up the GRAFLAB23 database structure. They are grouped by categories. They are found in the following directory on the Department 9735 HP735 server:

`/disk1/graflab23`

`diary`	Writes a log file to disk containing all text displayed on screen. This function is useful for documenting the results of a large verbose analysis for later review.
`eval`	Useful for performing batch processes using string matrices of curvenames.

`format`	Controls the display format of numbers. Example: Try `format short e`.
`prepepsf`	Prepares and prints an epsf file for import into word processor.
`startup`	Defines the path to find these GRAFLAB files.
`template`	A template file for writing new m files.

6.0 Programmer's Guide

Always include a "modified/date/who did it" comment when updating a GRAFLAB m-file to help with understanding future bugs caused by current features and fixes.

6.1 Functions that Call GRAFLAB System Routines

The routine SYS.M is designed to isolate the system calls to one routine, simplifying the process of porting the code to other operating systems. Six system calls are necessary in order to port the code.

1. LSD - List directories

2. LSF - List files

3. DELR - Recursive delete of directory and subdirectories underneath

4. CPR - Recursive copy of directory and contents

5. MV - Move a directory

6. MKDIR - Create a new directory

If this routine can be successfully ported to another operating system where MATLAB resides, then the rest of GRAFLAB can be transparently ported. Table 1 provides a list of the GRAFLAB routines that use SYS.M to make system calls, such as list files, delete, copy, move, or make directories.

Table 1. GRAFLAB Routines calling SYS.M

Routine Name	LSD	LSF	DELR	CPR	MV	MKDIR
`delcurf`			X			
`gcopy`	X	X		X		
`gdir`	X	X				
`gl2update`					X	
`ninput`						X
`savecurf`					X	X
`setd`					X	

Use the SYS command to do system commands

GRAFLAB has been written so that all the system commands are located in `sys.m` For example, to get a list (a dir in DOS) of all the *.inp files and read the text of each one, perform a task with the text from each file, execute the following commands.

LIST FILES

```
thelist=sys('lsf','*.inp');
```

```
if ~isempty(thelist)
    for j=1:size(thelist,1)
     theText=readtext(clearaws(thelist(j,:)));
     % do something here with the text
     % of each of the input files.
    end
end
```

• Note that `sys('lsf')` will list the files in `'./alldata'`.

• To check for the file existence, for example prior to opening it, use the following:

```
if ~isempty(sys('lsf','pathname/subdir/filename.txt')
    fid=fopen('pathname/subdir/filename.txt','rt');
else
    disp('WARNING:  file does not exist!')
end
```

LIST DIRECTORIES

Provides a directory of the directories.

```
theDirs=sys('lsd','pathname/filename');
```

• Note that `sys('lsd')` will list the directories in `'./alldata'`.

RECURSIVE DELETE

`sys('delr','./alldata/curvename')` will delete the `./alldata/curvename` directory and all subdirectories underneath it.

RECURSIVE COPY

`sys('cpr','./alldata/curvename','/disk1/username/project/alldata')` will copy the directory curvename, and all of its contents, under the second path.

RECURSIVE MOVE

`sys('mv','./alldata/curvename','/disk1/username/project/alldata')` will move the directory curvename, and all of its contents, under the second path.

MAKE DIRECTORY

`sys('mkdir','./alldata','curvename')` will make the directory curvename under ./alldata.

6.2 Add/Remove a Global Variable to GRAFLAB

The following four routines must be updated when adding a new global variable, and then *documented in each routine's* Help *file to show that it has been added.*

`actparm`	Returns the default value of the global when called.
`setdglob`	Calls `actparm.m` to set each of the globals (usually called at startup).
`setd`	Sets the new value of the global to an allowable value.
`showd`	Shows (and interprets in plain language) the value of the global.

Finally, `curvparm.m` is the routine that controls system variables.

6.3 GRAFLAB 23 Test Area

Located under the GRAFLAB directory is a subdirectory called `gl2test`. This directory contains sample data for testing GRAFLAB:

`Complex data`	A sample database to test GRAFLAB's handling of complex data.
`Neutral files`	From GRAFAID, 3 column, 2 column, bad curvenames (containing '.' or '-' or first char is not alpha).
`Universal files`	Universal files for input using uffinput.m.
`29E data`	Sample optical data files for input using input29ea.m and input29ed.m.
`Old Databases`	There is a database from version 1.0, 2.0 and 2.1 . These databases should be copied to a new area, and then used. DO NOT WORK DIRECTLY ON THESE DATABASES using the current version of GRAFLAB or you will update them and lose them.

6.4 Create a New GRAFLAB function

Located under the GRAFLAB directory is an m-file called `template.m`, which is the recommended best practice m-file header documentation.

```
function y=templdate(inputarg)
%      template.m
%     SYNTAX:
%     DESCRIPTION:
%     EXAMPLES:
%     SEE ALSO:
%     AUTHOR:          Date:
%     Modified:        Date:                Description:
```

7.0 MATLAB Memory Management

When processing large volumes of data, it is good practice to use the `clear xxx` utility to free memory stored in variable xxx that is no longer needed. This releases memory that MATLAB has allocated. However, for a variable that has been declared global, (`global xxx`), `clear xxx` clears only the active process (working space) variable and does NOT clear the global variable that was passed up to the parent MATLAB process. Therefore, MATLAB does not release the memory, it only makes it invisible to the active process.

To release memory from a global variable, perform a `clear global xxx`.

An additional useful command is `pack`, which defragments the memory that MATLAB occupies.

8.0 GRAFLAB Tips

This section contains a few handy "how to's" with examples.

1. Wildcard looping on SETD & SHOWD, for example, show curve titles of PA* curves:

```
a=gdir('PA*');

for  i=1:size(a,1);showd('ct',clearaws(a(i,:)));end
```

2. Check to determine whether vector A contains a scalar B

This takes advantage of the vectorized if statement in MATLAB.

```
A=[1 2 3 4 5];

B=[2];

if ~all(A-B)   % Uses vectorized if. If not all non-zero then
B is in A!

     disp('Yes, vector A contains at least one of scalar B!')

end
```

3. Using the eval command

You can evaluate string expressions as though they are GRAFLAB commands:

```
thecurve='PAGY';

a=['setd(''act'',' thecurve ')'];

eval(a)
```

The above three lines are the same as the following:

```
setd('act','PAGY')

Another example:

clist='curve';

setd('act',clist)

setdloc

x=eval(clist);
```

4. Multiply the y values of curve1 by a factor of 2 (This is an example of editing a GRAFLAB curve).

Simply set it active, make it available to the local space, then do the math and save only the data back out.

```
setd('act','curve1')

setdloc

curve1(:,2)=2*curve1(:,2);

savecurf('curve1',1,2)
```

5. Write select CURVE1 data to file named FILE1

Put the output of showd('cd' ...) into a file using writetext. The data can be truncated according to the independent variable, for example from TIME1 to TIME2.

```
writetext('FILE1',showd('cd','CURVE1',[TIME1   TIME2]));
```

6. Use graphical cursor input to create curves.

Create a global variable called test. Use ginput to digitize data points from the active figure window, scaling according to the axis scaling. Use help ginput for more complete instructions.

```
%  Need to have a plot figure displayed.

global test

[test(:,1)  test(:,2)]=ginput(5);

% digitize the data using the cursor

savecurf('test',0,6)
```

Appendix A
Example Startup.m file

This shows an example startup.m file. The startup.m file is executed automatically upon startup by MATLAB. Five key functions are accomplished by this file. Items 1, 3, and 4 are necessary for GRAFLAB to operate properly.

1. Set the appropriate paths.
2. Echo the version of GRAFLAB.
3. Set the GRAFLAB globals (`setdglob`).
4. Check to determine whether the database needs updating; if so, perform a `gl2update`.
5. Customize the user's local GRAFLAB environment.

```
% EXAMPLE STARTUP.M FILE

a=['/matlab_42a/toolbox/matlab/general:'];
a=[a  '/matlab_42a/toolbox/matlab/ops:'];
a=[a  '/matlab_42a/toolbox/matlab/lang:'];
a=[a  '/matlab_42a/toolbox/matlab/elmat:'];
a=[a  '/matlab_42a/toolbox/matlab/specmat:'];
a=[a  '/matlab_42a/toolbox/matlab/elfun:'];
a=[a  '/matlab_42a/toolbox/matlab/specfun:'];
a=[a  '/matlab_42a/toolbox/matlab/matfun:'];
a=[a  '/matlab_42a/toolbox/matlab/datafun:'];
a=[a  '/matlab_42a/toolbox/matlab/polyfun:'];
a=[a  '/matlab_42a/toolbox/matlab/funfun:'];
a=[a  '/matlab_42a/toolbox/matlab/sparfun:'];
a=[a  '/matlab_42a/toolbox/matlab/plotxy:'];
a=[a  '/matlab_42a/toolbox/matlab/plotxyz:'];
a=[a  '/matlab_42a/toolbox/matlab/graphics:'];
a=[a  '/matlab_42a/toolbox/matlab/color:'];
a=[a  '/matlab_42a/toolbox/matlab/sounds:'];
a=[a  '/matlab_42a/toolbox/matlab/strfun:'];
a=[a  '/matlab_42a/toolbox/matlab/iofun:'];
a=[a  '/matlab_42a/toolbox/matlab/demos:'];
a=[a  '/matlab_42a/toolbox/local:'];
a=[a  '/matlab_42a/toolbox/signal:'];
a=[a  '/disk1/graflab23:'];
a=[a  '/disk1/glutil/gmechanical:'];
a=[a  '/disk1/glutil/gthermal:'];
a=[a  '/disk1/glutil/gshaker'];
a=[a  '/disk1/glutil/testcases'];
path(a);
clear a
disp(['*******************************************'])
```

```
 disp(['    Welcome to GRAFLAB version ' curvparm('GL_VER')])
 disp(['             for ' curvparm('MACHINE_ID')])
 disp(['****************************************'])
 disp(' ')
 disp(' Type buglist at the prompt to get a current bug
listing')
%
% SET THE GRAFLAB GLOBALS
 setdglob
 % UPDATE THE DATABASE IF NECESSARY
gl2update
% OPTIONAL, CUSTOMIZED STUFF
 setd('dg','on')    % TURN ON GRIDS
 setd('tj','center')   % CENTER THE TITLE
```

Index

Distribution

2	MS 0415	W. N. Dunn Jr.	1	MS 0439	R. V. Field Jr.
5	MS 0436	G. L. Maxam	1	MS 0439	C. W. G. Fulcher
1	MS 0437	H. S. Morgan	1	MS 0439	D. W. Lobitz
1	MS 0437	C. R. Adams	1	MS 0439	G. M. Reese
1	MS 0437	J. B. Aidun	1	MS 0439	D. J. Segalman
1	MS 0437	J. G. Arguello Jr.	1	MS 0439	H. P. Walther
1	MS 0437	S. W. Attaway	1	MS 0439	W. R. Witkowski
1	MS 0437	M. L. Blanford	1	MS 0555	D. B. Davis
1	MS 0437	K. H. Brown	1	MS 0555	M. S. Garrett
1	MS 0437	S. N. Burchett	1	MS 0555	V. I. Bateman
1	MS 0437	R. S. Chambers	1	MS 0555	D. L. Gregory
1	MS 0437	A. F. Fossum	1	MS 0557	T. J. Baca
1	MS 0437	J. D. Gruda	1	MS 0557	P. S. Barney
1	MS 0437	M. W. Heinstein	1	MS 0557	T. G. Carne
1	MS 0437	E. L. Hoffman	1	MS 0557	S. E. Klenke
1	MS 0437	J. Jung	1	MS 0557	J. P. Lauffer
1	MS 0437	S. W. Key	1	MS 0557	R. L. Mayes
1	MS 0437	J. R. Koteras	1	MS 0557	C. C. O'Gorman
1	MS 0437	C. S. Lo	1	MS 0557	T. L. Paez
1	MS 0437	F. J. Mello	1	MS 0557	M. J. Sagartz
1	MS 0437	K. E. Metzinger	1	MS 0615	R. L. Perry
1	MS 0437	J. A. Mitchell	1	MS 0865	J. L. Moya
1	MS 0437	M. K. Neilson	1	MS 0865	J. R. Barnum
1	MS 0437	J. Pott	15	MS 0865	J. S. Cap
1	MS 0437	E. D. Reedy Jr.	2	MS 0865	J. E. C de Baca
1	MS 0437	L. A. Schoof	1	MS 0865	T. Y. Chu
1	MS 0437	G. D. Sjaardema	2	MS 0865	R. D. Foral
1	MS 0437	J. R. Stewart	2	MS 0865	W. Gill
1	MS 0437	C. M. Stone	1	MS 0865	H. G. Hudson
1	MS 0437	J. W. Swegle	2	MS 0865	D. O. Smallwood
1	MS 0437	M. R. Tabbara	1	MS 0865	J. E. Solberg
1	MS 0437	L. M. Taylor	1	MS 1135	J. R. Garcia
1	MS 0437	B. J. Thorne	1	MS 1135	R. G. Coleman
1	MS 0437	G. W. Wellman	1	MS 1135	N. T. Davie
1	MS 0439	D. R. Martinez	1	MS 1135	T. C. Togami
1	MS 0439	K. F. Alvin	1	MS 1392	V. Gabbard
1	MS 0439	J. L. Dohner	1	MS 0439	D. B. Longcope
1	MS 0439	C. R. Dohrmann	1	MS 0439	J. R. Red-Horse
1	MS 0439	B. Driessen	1	MS 0439	J. M. Redmond
1	MS 0439	M. S. Eldred			

1	MS 0557	T. W. Simmermacher
1	MS 9018	Central Technical Files 8940-2
2	MS 0899	Technical Library 4916
2	MS 0619	Review and Approval Desk 12690 For DOE/OSTI