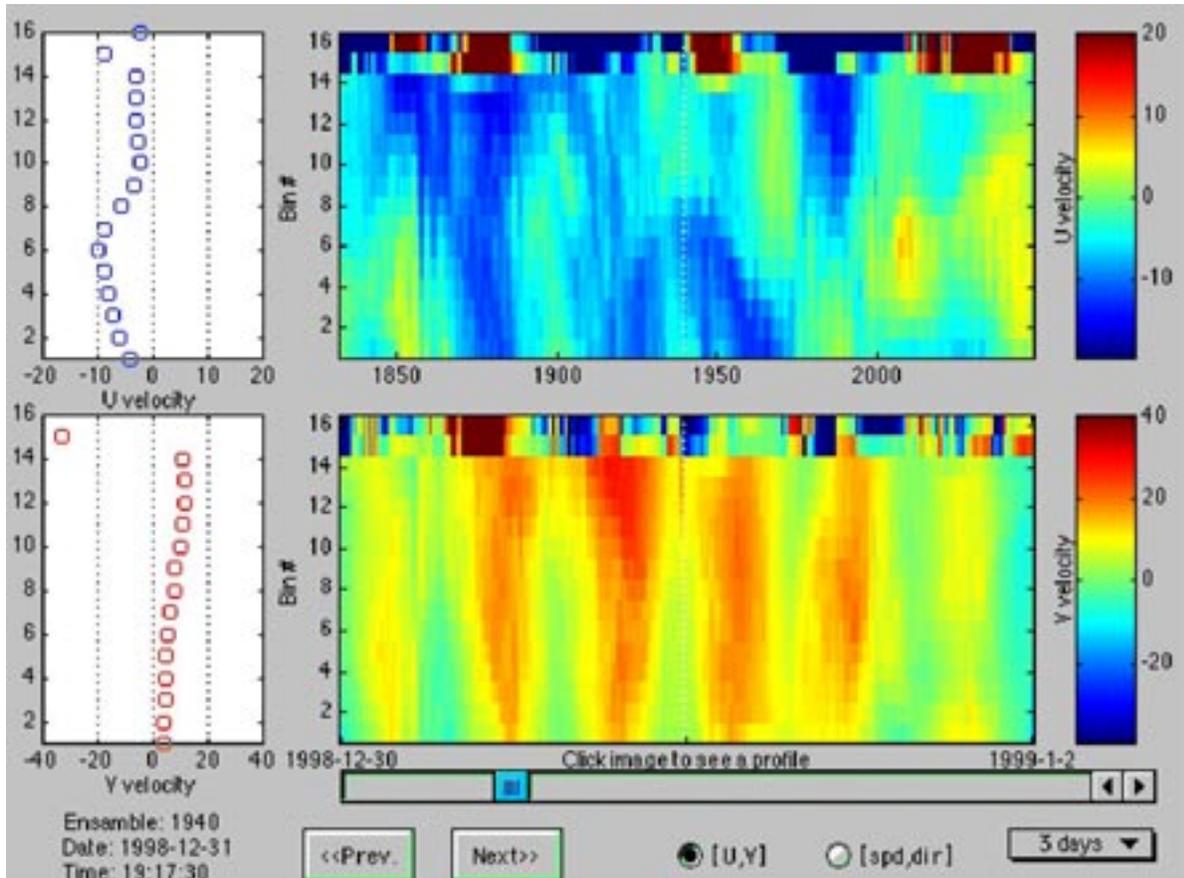


# CMGTool User's Manual

## Open-File Report 02-19



Jingping Xu, Fran Lightsom, Marlene Noble, Charles Denham

Coastal and Marine Geology Program  
U. S. Geological Survey  
Menlo Park, CA 94025

Any use of trade, product, or firm names is for descriptive purposes only and does not imply endorsement by the U.S. Government.

## TABLE OF CONTENTS

1. OVERVIEW.....	2
2. INSTALLATION2.....	2
3. USING CMGTooL GUI.....	3
3.1 Starting CMGTooL GUI.....	3
3.2 Loading NetCDF files.....	4
3.3 The CMGTooL Window.....	5
3.4 Time-Domain Functions.....	6
3.4.1 AnalyzeIt – The time-series analyzing methods.....	6
3.4.2 GraphIt – graphic presentation of time-series variables.....	9
3.4.3 SaveIt – Saving variables into new files.....	13
3.5 Frequency-Domain Functions.....	13
3.5.1 AnalyzeIt – The time-series analyzing methods.....	14
3.5.2 GraphIt – graphic presentation of spectra.....	15
3.5.3 SaveIt – Saving variables into new files.....	16
4. CMGTooL LIBRARY.....	16
5. PROSPECT IN MATLAB.....	25
5.1 Data files.....	26
5.2 Building the data matrix with cmgfbuild.....	26
5.3 Pre-processing the data matrix.....	27
5.4 Computing spectral properties using SPECACR.....	27
5.5 Generating spectral plots using SPECPL.....	28
6. ACKNOWLEDGEMENT.....	29
7. REFERENCE.....	29

## 1. OVERVIEW

During the past several years, the sediment transport group in the Coastal and Marine Geology Program (CMGP) of the U. S. Geological Survey has made major revisions to its methodology of processing, analyzing, and maintaining the variety of oceanographic time-series data. First, CMGP completed the transition of its oceanographic time-series database to a self-documenting NetCDF (Rew et al., 1997) data format. Second, CMGP's oceanographic data variety and complexity have been greatly expanded from traditional 2-dimensional, single-point time-series measurements (e.g., Electro-magnetic current meters, transmissometers) to more advanced 3-dimensional and profiling time-series measurements due to many new acquisitions of modern instruments such as Acoustic Doppler Current Profiler (RDI, 1996), Acoustic Doppler Velocimeter, Pulse-Coherence Acoustic Doppler Profiler (SonTek, 2001), Acoustic Backscatter Sensor (Aquatec, 2001). In order to accommodate the NetCDF format of data from the new instruments, a software package of processing, analyzing, and visualizing time-series oceanographic data was developed. It is named CMGTool.

The CMGTool package contains two basic components: a user-friendly GUI for NetCDF file analysis, processing and manipulation; and a data analyzing program library. Most of the routines in the library are stand-alone programs suitable for batch processing.

CMGTool is written in MATLAB computing language (The Mathworks, 1997), therefore users must have MATLAB installed on their computer in order to use this software package. In addition, MATLAB's Signal Processing Toolbox is also required by some CMGTool's routines. Like most MATLAB programs, all CMGTool codes are compatible with different computing platforms including PC, MAC, and UNIX machines (Note: CMGTool has been tested on different platforms that run MATLAB 5.2 (Release 10) or lower versions. Some of the commands related to MAC may not be compatible with later releases of MATLAB)

The GUI and some of the library routines call low-level NetCDF file I/O, variable and attribute functions. These NetCDF exclusive functions are supported by a MATLAB toolbox named NetCDF, created by Dr. Charles Denham (<http://crusty.er.usgs.gov/~cdenham>). This toolbox has to be installed in order to use the CMGTool GUI.

The CMGTool GUI calls several routines that were initially developed by others. The authors would like to acknowledge the following scientists for their ideas and codes: Dr. Rich Signell (USGS), Dr. Chris Sherwood (USGS), and Dr. Bob Beardsley (WHOI).

Many special terms that carry special meanings in either MATLAB or the NetCDF Toolbox are used in this manual. Users are encouraged to read the documents of MATLAB and NetCDF for references.

## 2. INSTALLATION

The file structure of CMGTool package is shown in the directory tree (Figure. 1). The GUI components are stored in a directory named **cmgtool**, under which there are 5 subdirectories (**corefiles**, **rich**, **t\_tide**, **chuckstats**, and **gapfill**). The **cmgmfiles** directory contains the stand-alone routine library. The installation of the CMGTool package is fairly straightforward. First, create two separate directories on the hard disk, one for the GUI and one for the library. Then copy the files, including the subdirectories, to their appropriate directory. Finally modify the MATLAB search path to include these directories and subdirectories.

The installation of the NetCDF toolbox is a little more complicated. See <http://crusty.er.usgs.gov/~cdenham> for details.

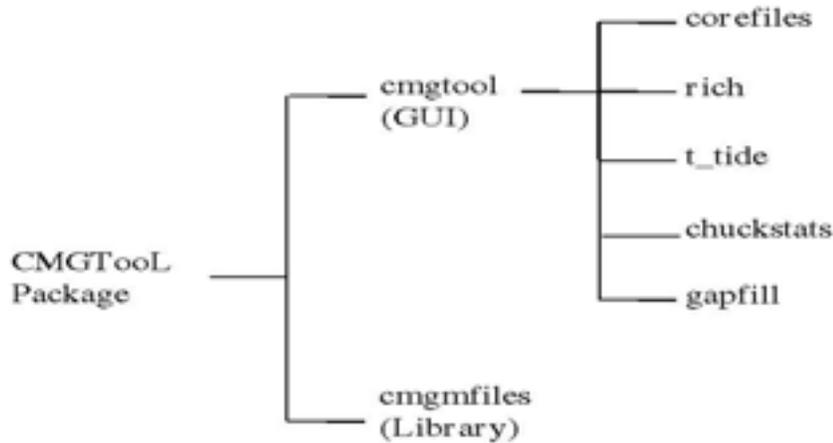


Figure 1. Directory tree showing the file structure of the CMGTool package. The MATLABPATH needs to be modified to include all directories and subdirectories in the tree.

### 3. USING CMGTool GUI

The primary purpose of the CMGTool GUI is to provide a user-friendly oceanographic data analysis and processing package utilizing MATLAB's powerful routines for scientists who may only have limited knowledge of MATLAB. CMGTool GUI is operated almost entirely by mouse-clicks, with keyboard input limited to only few situations. This section describes the main functions of CMGTool GUI and how these functions are used.

#### 3.1 Starting CMGTool GUI

The CMGTool GUI is started by typing *cmgtool* in the MATLAB Command Window (Figure 1). Notes: all screen-capture images in this document are derived from a MAC. They might look different on either PCs or UNIX computers.

The GUI utilizes both pushbuttons as shown in Figure 2 and a menu bar (not shown) that allows additional functions. The **LOAD A FILE** button is used to load a NetCDF file. Clicking the **RESET** button will clear all loaded files from both the GUI window and the MATLAB memory. The **QUIT** button allows user to quit the program and clear all relevant variables (mainly global variables) from the memory. Two menus, wrapped with square brackets, are added to the built-in MATLAB menu bar. The **[Tools]** menu allows user to choose between programs that analyzing/processing data in the Time Domain (default) or Frequency Domain. After a NetCDF file is loaded, the title of the CMGTool window will change accordingly depending on user's selection from the **[Tools]** menu. The **[Help Tips]** menu provides brief information for most of the commonly used GUI objects. The help information of the chosen item appears in a textbox at the lower left corner of the window. This textbox disappears when the user clicks anywhere inside the GUI window.

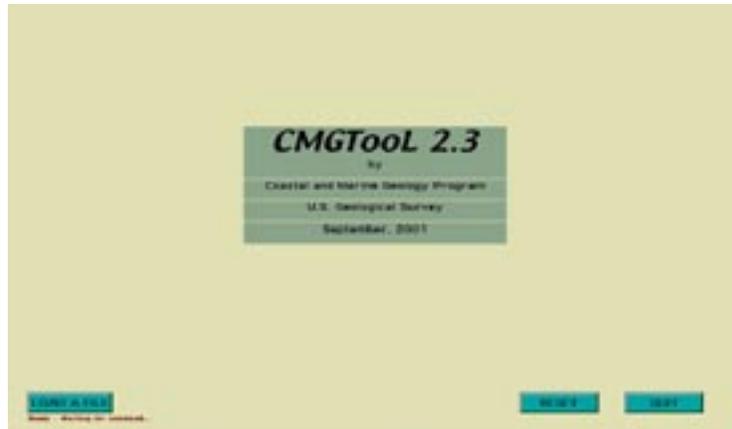


Figure 2. The start-up window of the CMGTool GUI. If filenames are provided in the typed command, e.g. `cmgtool xyz.nc` or `cmgtool({'abc.nc','xyz.nc'})`, this display will not be shown.

Users may also choose to start the CMGTool GUI with an argument: `cmgtool(filename)`, where filename is a string when there is only one file e.g. '5103-a.nc', or a cellstring when there are multiple files e.g. {'5103-a.nc','5104-a.nc','adcp3.cdf'}. The file(s) in the argument will be loaded at the start of CMGTool. Note: The CMGTool GUI requires that all NetCDF files must have a time variable named 'time' in true Julian days. If the file is EPIC compliant (Soreide, 1997), there should be two time variables ('time' in true Julian days and 'time2' in milliseconds).

### 3.2 Loading NetCDF files

Clicking the **LOAD A FILE** button in Figure 2 will bring in the file selection I/O interface from which users may select a NetCDF file to load. PC and UNIX users may choose to only display files with .cdf (or .nc) suffix. Users will be warned if any error occurs during file loading. As soon as a file is loaded, The CMGTool GUI window appears on the screen (Figure 3).

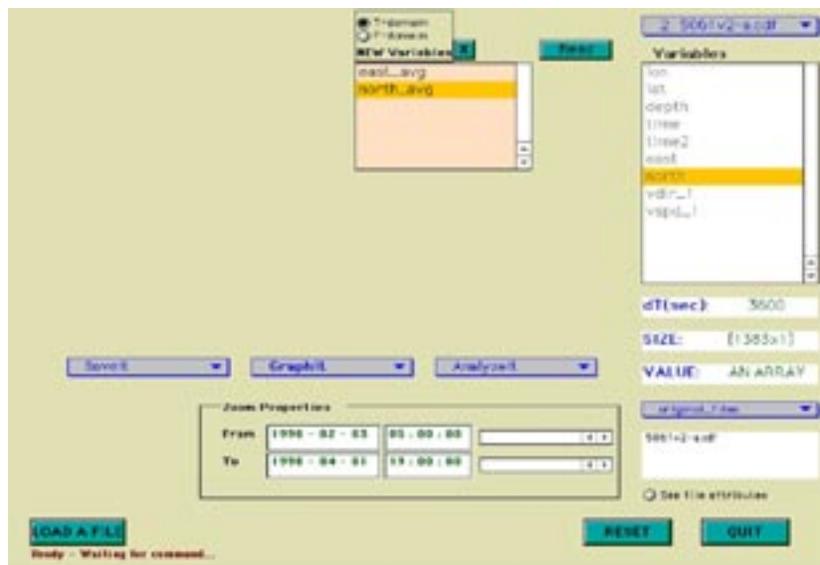


Figure 3. The screen image of a fully-functional CMGTool GUI.

### 3.3 The CMGTool Window

As soon as a file is loaded the CMGTool becomes fully functional and most GUI objects (pushbuttons, listboxes, popupmenus etc.) are displayed in the CMGTool window (Figure 3). As many as 9 files can be loaded into one session.

Most GUI objects are self-describing. The popup menu at the upper right corner lists all the loaded files from which users can choose for processing, plotting, or analysis. The **Close File** push button to the left allows users to close the selected file and remove its variables and attributes from the memory. When a file is chosen, all variables in the file are displayed in the **Variables** listbox at the right side of the GUI window. The time increment, in seconds, is displayed in the **dT** textbox below the listbox. The beginning and ending time of the file are displayed in the **Zoom Properties** frame. The global attributes of the file can now be viewed by selecting from the **ATTRIBUTES** popup menu. The **See file attributes** radio button should be in the active state (down position) to view a global attributes. Therefore global attributes can be viewed at any time by pressing the radio button.

When a variable in the listbox is clicked, its size and value are displayed. If the variable is an array, 'AN ARRAY' is displayed in the **VALUE** textbox. The attributes of the selected variable can be viewed by selecting from the **ATTRIBUTES** popup menu. The data values of a selected variable may be examined by clicking the **Read** button to the left of this listbox.

The **New Variables** listbox (Figure 3) is used to store new variables calculated from the analysis methods that will be described later. An **X** push button is activated when a new variable is clicked and can be used to remove the chosen new variable from the workspace. The time increment, size, and attributes of the new variables are displayed when they are selected. When users click a variable in either listbox, this listbox becomes highlighted with a different background color. The text color in the **dT**, **SIZE**, and **VALUE** textbox also changes when a different listbox is highlighted. The contents of each variable from the **New Variables** listbox may be examined by clicking the **Read** button.

The **T-domain** and **F-domain** radio buttons above the **New Variables** listbox allow users to retrieve variables created from either domain without changing to that domain using the **[Tools]** menu. When **T-domain** is activated, the **New Variables** listbox will display the new variables created by the time-domain methods, and vice versa. In another word, users can access the new time-series variables from frequency-domain desktop by activate the **T-domain** radio button.

The three popup menus in the middle of the window contain the main functions of the CMGTool GUI. **AnalyzeIt** allows users to choose from different analyzing methods for processing a variable or variables displayed in both variable listboxes. These variables can be derived from different files. **GraphIt** allows users to create a variety of plots of the variables. **SaveIt** allows users to save selected variables into a file that may be one of three formats: NetCDF, ASCII text, and MATLAB binary. The programs listed under the **AnalyzeIt** and **GraphIt** change when users switch from Time Domain to Frequency Domain in the **[Tools]** menu.

### 3.4 Time-Domain Functions

#### 3.4.1 AnalyzeIt – The time-series analyzing methods

In order to use these methods on time-series data, users are first required to provide input parameters. To input a variable, users can either first choose a variable from either listbox and then click the << **Variables** button, or directly type the name of the variable in the textbox. The

first input method is recommended to avoid typos. After all required input parameters are filled, click the **RUN** button to execute. Most outputs are new variables that are shown in the **New Variables** listbox. An attribute of each new variable indicates its origin. Some methods do not create new variables. Instead, the output goes directly to the MATLAB Command Window. So far, the following analyzing/processing methods are implemented in **AnalyzeIt**.

- **Arithmetic:** This method allows freehand input of a simple arithmetic formula in the textbox. For example,  $mylunch = 30*eggs + 2*(chickens./wings)$ , in which *eggs*, *chickens*, and *wings* are variables found in either variable listbox. *mylunch* is a new variable that will be saved in the **New Variables** listbox. This method can only deal with variables from a single loaded file plus the new variables. New variables generated from this method carry the ‘\_fh’ suffix. (Note – MATLAB syntax (e.g. ‘./’) needs to be used)
- **BandPass:** This method is mostly used to lowpass time-series data. It call the `filtfilt.m` from MATLAB’s signal processing toolbox. Sampling frequency and cut-off frequency (both in cycle/day) are required. Users may also choose to resample the filtered time-series by providing a subsample interval (e.g., 6 for every sixth point). Typing *all* in the input will process all variables in the current file that have the length of the timebase. An error will occur if all variables don’t have the same dimension. New variables generated from this method carry the ‘\_filt’ suffix.
- **LowPass:** This method is used to remove tidal signals from time-series data. Hourly data is required as input. Users may also choose to resample the filtered time-series by providing a subsample interval (e.g., 6 for every sixth point). Typing *all* in the input will process all variables in the current file that have the length of the timebase. New variables generated from this method carry the ‘\_f33’ suffix.
- **Rotate:** This method is used for vector rotation in a Cartesian coordinate. Counterclockwise angles are positive. New variables generated from this method carry the ‘\_r’ suffix.  
Input: U, V  
Output: U\_r, V\_r
- **uv2speed:** This method calculates speed and direction from the two orthogonal velocity components (e.g., east and north). Zero degree is true north. New variables generated from this method are `spd_EN` and `dir_EN`.  
Input: U, V  
Output: `spd_EN`, `dir_EN`
- **speed2uv:** This method calculate the East and North velocity components from speed and direction. Directions are in degrees (using oceanographic convention, i.e. 0-North, 90-East, 180-South, 270-West). New variables generated from this method are `EW_vel` and `NS_vel`.  
Input: `spd`, `dir`  
Output: `EW_vel`, `NS_vel`
- **Average:** This method computes the point average of time-series data (e.g. create hourly averaged data from a time-series that is sampled in higher frequency). Typing *all* in the input will process all variables in the current file that have the length of the timebase. New variables generated from this method carry the ‘\_avg’ suffix.

- **Subsample:** This method is used to subsample time-series data. Typing *all* in the input will process all variables in the current file that have the length of the timebase. New variables generated from this method carry the ‘\_sub’ suffix.
- **Merge/Interp:** This method carries out two operations: 1) to merge a variable ( or variables) from two or more different files; and 2) to interpolate a variable or variables to a different a different timebase. In the **Merge** operation, the variables must exist in all files that are being merged. Data gaps or overlaps may be present. The time increments of the files may be different, but the time increment of the merged new variables depends on the time-interval provided by the user. When a variable or file number is entered, the Tstart and Tstop fields are automatically filled. The File numbers and Time interval (sec) are required inputs. Users may also change the Tstart and/or Tstop. In the **Interp** operation, only the variables from the same file may be interpolated therefore the File numbers field is not needed. Users are required to choose an interpolation method from the popup menu and to provide a new time increment. New variables generated from this method carry the ‘\_merg’ suffix.
- **Inter-file calc:** This method performs arithmetic calculations fo two variables from two different files. At present, only addition and subtraction are implemented. If the time increments of the two files are different, a new timebase with the larger value of the time increments is created for the resultant new variable. New variables generated from this method carry the ‘\_f2f’ suffix.
- **Principal axis:** The method computes the major and minor axis properties from a vector time-series. The principal axes as well as the means are plotted in a figure that also includes the scatter plot and ellipse. There is no new variable output.
- **Wind stress:** This method calculates wind stress from given wind speed and direction. The program first estimates the wind speed at 10 meter above sealevel using an empirical iteration scheme (Wu, 1980). Then it calculates the East and North components of wind stress. New variables are identified as ‘Wstress.EW’ and ‘Wstress.NS’.
- **Bridge:** This method fills gaps in a time-series data with linear (for short gaps) or spectral (for long gaps) approach based on user’s instruction. It works extremely well for periodic data (e.g., tide). Typing *all* in the input will process all variables in the current file that have the length of the timebase. Gap information is printed in the MATLAB Command Window. New variables are identified with a ‘\_brdg’ suffix..
- **Statistics:** This method calculates the error bar of a time-series in addition to the mean, standard deviation, maximum, and minimum. There is no new variable and all the output is printed in the MATLAB Command Window. Users are required to either provide the Degrees of Freedom (DOF) or to select the value of the autocorrelation scale from a plot, using the mouse pointer. The DOF is length of the record divided by the autocorrelation scale (Emery and Thomson, 1997; Noble and Ramp, 2000). For ADCP data, users are asked to select one bin to be used for DOF estimate. Also, users may choose to run the statistics on a portion of the time-series by setting the start and end times from the sliders or typing in the start and end dates/times in the From and To textbox (Figure 3). Typing *all* in the variable input textbox will process all variables in the current file that have the length of the timebase. Note: Users may choose to use the subtidal (filtered with **Filter-PL33**) data to calculate error bars when dealing with hourly data.

- **Tidal Analysis:** This method employs the T\_Tide package (<http://www2.ocgy.ubc.ca/~rich>) to perform the Forman tidal analysis (Forman, 1978). If there are two entries in the Variables input field, a complex time series ( $U+iV$ ) is formed and tidal analyses are performed on this complex data. In addition to filling the Variables field, users may also need to fill the Tstart and Latitude fields (using the format given) for node correction. No new variables are produced. All outputs are displayed in the MATLAB Command Window. They include frequency of tidal constituents (cph), constituent major and minor axes, 95% confidence intervals for major and minor axes, ellipse orientations (degrees), 95% confidence intervals for ellipse, constituent phases (degrees relative to Greenwich), and 95% confidence intervals for phase. Users can choose to output the O1, K1, M2, and S2 only. A plot is also displayed to compare the original and predicted values.

### 3.4.2 GraphIt – Graphic presentation of time-series data

This popup menu contains the functions to make plots or displays. Users are first required to provide input parameters. To input a variable, users can either first choose a variable from either listbox and then click the << **Add Var.** button (the name of the button might be different for different plots) button, or directly type the name of the variable in the textbox. The first input method is recommended to avoid typos. Using the **[Save As]** menu that is added to the MATLAB default menu of each plot, the plots or displays may be saved in different graphic formats (e.g. JPEG, POSTSCRIPT, ILLUSTRATOR), or sent directly to a printer. The following plot types have been implemented:

- **Time-series plot:** Users can plot time-series data into as many as 4 panels (subplots). Each panel may have one or multiple variables that can come from different files (Figure 4). To make a vector (stick) plot, use square brackets around the two velocity components, e.g. [U, V]. For most variables the Title and Ylabel fields are automatically filled, but users may

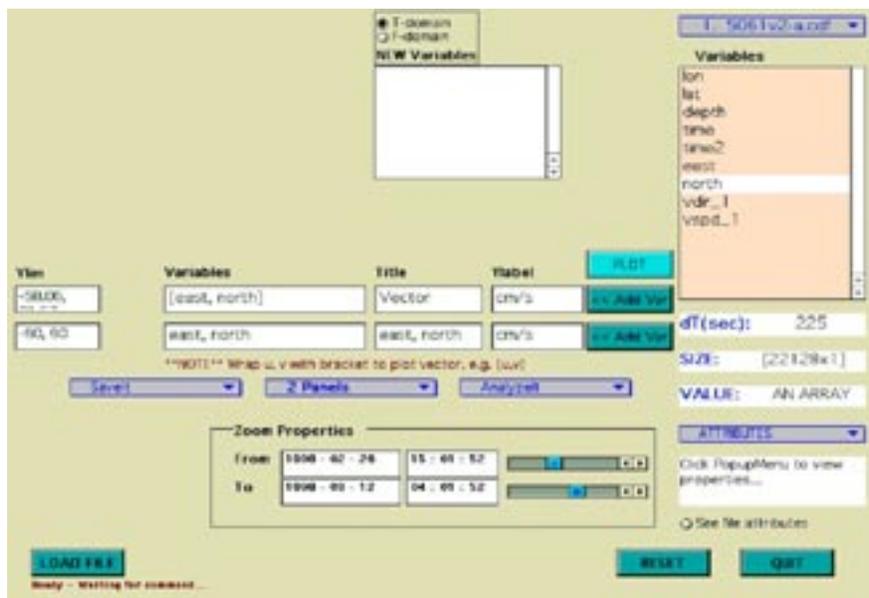


Figure 4. The CMGTool GUI window that shows the process of plotting a time-series data. The two panels have the same two variables (east and north) from the **Variables** listbox. But the variables in the top panel are wrapped by square brackets to have a stick plot. Notice that the sliders have been adjusted in order to zoom in the plot.

change them if desired. Users have an option of turning the legend on and off. After the initial plot is created, users may change the Ylim (y-axis limits) for each subplot. Users may also zoom in on the plot by adjusting the sliders or typing in the date/times in From and To textboxes inside the Zoom Properties frame. When plotting ADCP variables, users are asked to input the number of bins whose records are to be plotted. Normally a one-panel plot is recommended for an ADCP variable. If users need to plot multiple variables, but in the same bin, then multiple-panels should be used and each variable should be in a separate panel. For burst-type data such as ADVs and Electro-magnetic current meters, users are asked to provide the burst number(s) to be plotted. The starting time of each burst is shown at the lower left corner of each subplot, and the x-axis shows the number of seconds from the start of each burst. In each subplot, users may use the right mouse button (Option + click for Macintosh) to change line colors or line types. An example of time-series plot is shown in Figure 5.

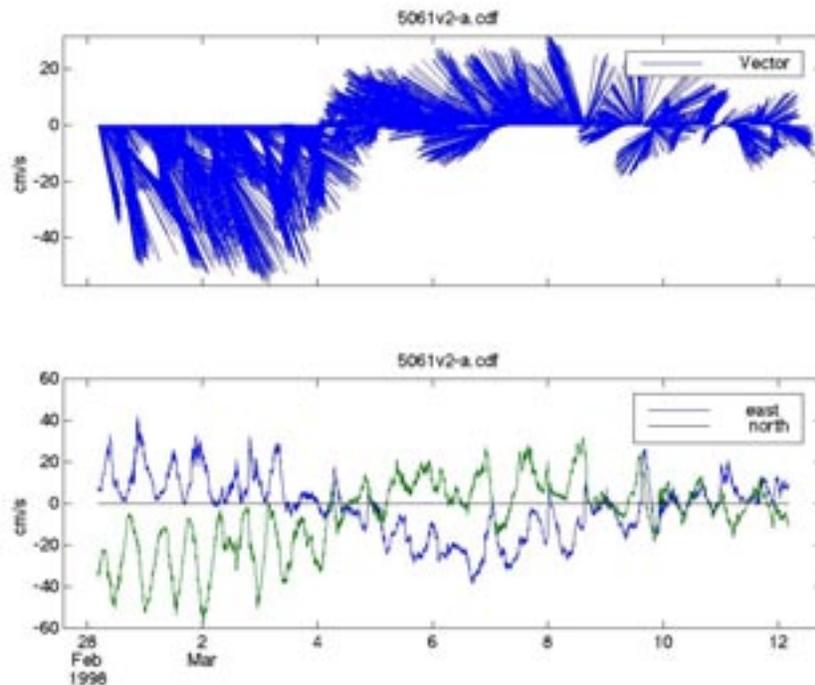


Figure 5. The time-series plot that is created from the settings shown in Figure 3. The lower panel shows the lineplot of the two velocity components. The top panel is the vector (stick) plot of the same time-series.

- **Scatter/Progressive Vector plots:** Scatter plots are very useful in determining the tidal ellipse and as well as the principal axis of winds or tidal/subtidal currents. Users can choose to plot the ellipse only without the data points. The axes limits can also be changed. The program requires a vector input (East and North components). If the time-series is too long only a portion of the data points are plotted for graphic clarity. The properties of the principal axis are also plotted (Figure 6). The Progressive Vector plots display the history, in a spatial sense, of a time-series vector (Figure 6). Time intervals are also calculated and marked on the plot. It also requires a vector input (e.g. the East and North components).

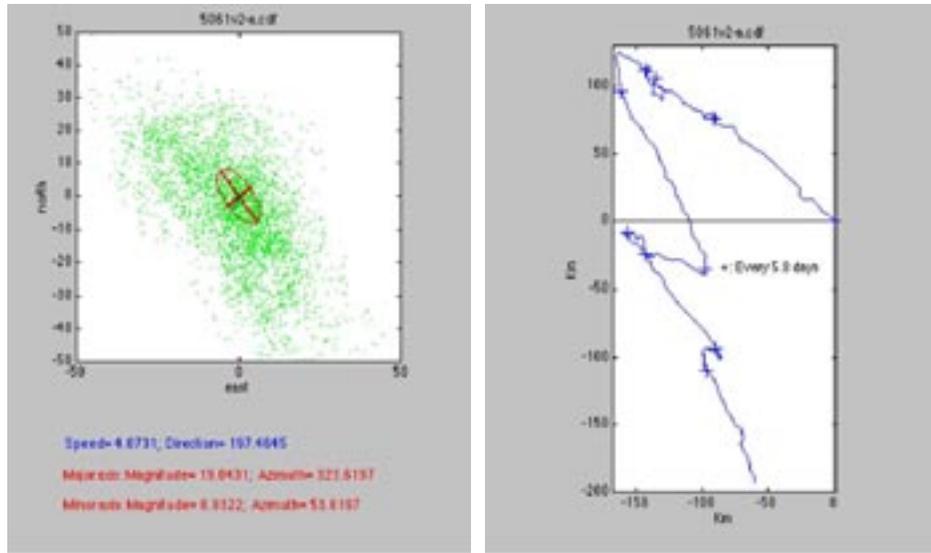


Figure 6. Examples of a scatter plot (left) and a progressive vector plot (right). The principal axis ellipse and its parameters are also shown in the scatter plot. In the progressive vector plot, the '+' symbols are time markers.

- **ADCPVIEW:** This is a tool to examine ADCP data whose dimension must be  $[a \times b]$  and  $b > 1$  (Figure 7). Required inputs are the two orthogonal, horizontal components of the ADCP measurements. Users may select to display  $[u,v]$  or  $[\text{speed}, \text{direction}]$ . From the popmenu at the lower right corner users can set the time span of the display to 1, 3, 7, and 30 days or the whole record. Use the slider to change the time segment of the display. Click anywhere in either image to get vertical profiles of the data at that specific time, these profiles are plotted in the left side of the window. The date, time, and ensemble number are displayed below the profile plots. Use <<Prev. and Next>> buttons to move one ensemble at a time.

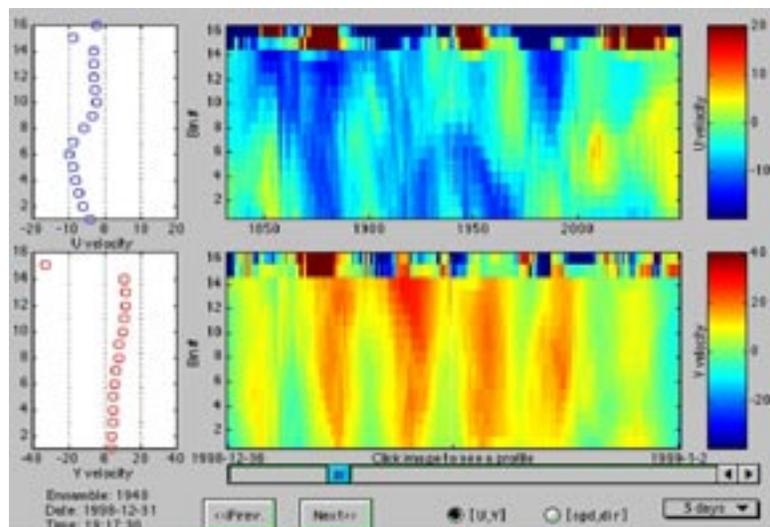


Figure 7. An ADCPVIEW screen shot. The scales of the color images are shown in the two color bars to the right. Note that the top two bins of this particular data are not reliable due to water surface scattering.

- Histogram:** This creates a histogram plot of time-series data (Figure 8). Users have options of plotting one or multiple variables at a time. There are 20 bins (default) in the horizontal axis but this can be changed by user. When plotting ADCP or ADV data, only one variable may be plotted at a time, and users are asked to provide the bin/burst number that is to be plotted.

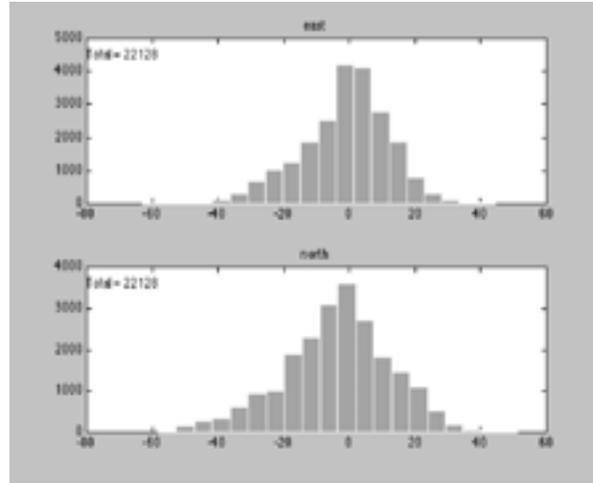


Figure 8. Example of a histogram plot. The total number of point is printed at the upper left corner of each panel.

- Polar:** This creates a polar plot (Figure 9). The input data can be: 1) an orthogonal pair (U, V); or 2) the speed and direction of the time-series measurements. The oceanographic convention (North=0, East=90, South=180, West=270) is used.

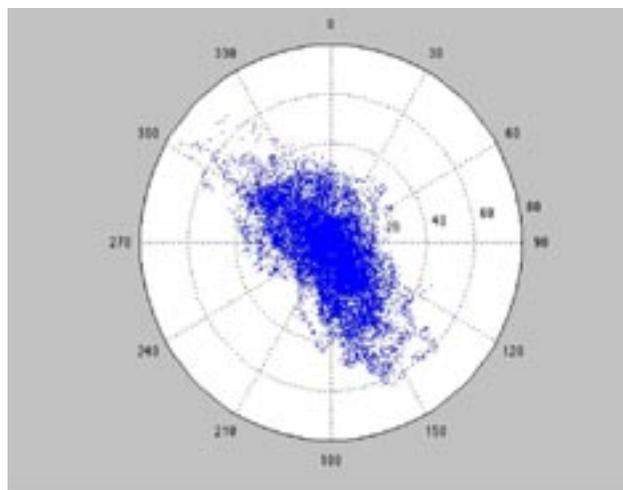


Figure 9. Example of a polar plot of the same data shown in Figure 8.

### 3.4.3 SaveIt – Saving variables into new files

The **SaveIt** popup menu provides users with three formats of saved files: **ASCII**, **MATLAB**, and **NetCDF**. A listbox named **SAVE Variables** is displayed at the upper left corner along with five push buttons that allow user to add variables into the listbox, remove variable from the listbox, rename variables, and save the variables into a file. ASCII allows users to save variables into text files. Some file and variable attributes are saved into the headers of the text file. The first six columns hold the time variable in [yyyy mm dd hh mm ss] format, with the variables saved in subsequent columns. Choosing MATLAB allows users to save variables into a MATLAB binary (\*.mat) file, and NetCDF allows users to save variables into a NetCDF file. Variables from different files that have different data lengths or time increments can be saved into a single output file. However, users are asked to form a common timebase using either the largest or smallest time increment; and the variables are interpolated to the common timebase if necessary. Users are also asked if they need to save a certain time range instead of the whole record (default). If the answer is yes then users are lead back to the CMGTool window to use the sliders to specify the time range. The time variable(s) is always automatically saved. ADCP or ADV variables that are matrices cannot be saved into NetCDF files on Macintosh computers owing to technical difficulties. Figure 10 shows the CMGTool window when ASCII is chosen.

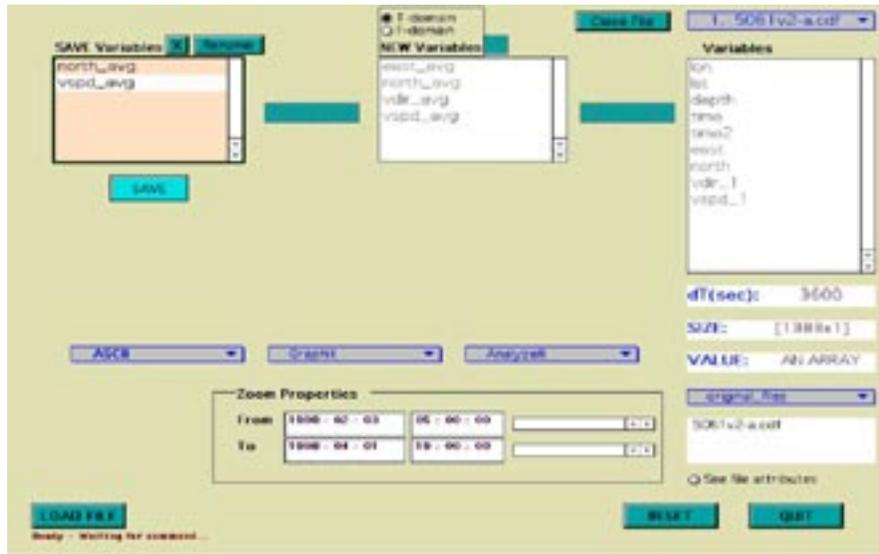


Figure 10. The screen shot of the GUI when an ASCII file is being saved. Variables from both the **Variables** and **New Variables** listboxes can be saved together into a new file. After the **SAVE** button is clicked the first time, users can change the time range from the sliders that are now activated.

### 3.5 Frequency-Domain Functions

CMGTool always starts in Time Domain. Use the [**Tools**] menu to switch to Frequency Domain. The title of the CMGTool GUI window changes to **CMGTool – Spectral** after the domain is switched. The new variables that have been saved in the Time Domain will be hidden. The entries under the **AnalyzeIt** and **GraphIt** also change accordingly.

#### 3.5.1 AnalyzeIt – Spectral analysis of time-series data

In order to use these methods on time-series data, users are first required to provide input parameters. To input a variable, users can either first choose a variable from either listbox and then click the << **Variables** button, or directly type the name of the variable in the textbox. The first input method is recommended to avoid typos. After all required input parameters are filled,

click the **RUN** button to execute. The output spectra are saved as new variables in the **New Variables** listbox. An attribute of each new variable indicates its origin.

**Spectrum** is the only method in the **AnalyzeIt** popup menu. This function is used to estimate auto- and cross-spectrum of the variable(s). For cross-spectrum, the two variables must be enclosed in the square brackets, e.g. [var1, var2], in the variable field (Figure 11). In order to run spectra on a new variable that was created in the Time Domain, users need to click the **T-domain** radio button to have those variables displayed in the **New Variables** listbox. The four parameter fields on the left are editable textboxes. The **Window** and **Detrending** popup menus provide users with different options of windows (e.g. Hanning, Cosine, etc) and detrending methods. Auto-spectrum is identified by the ‘\_spec’ suffix in the new variable name and has a dimension of [a x 5]. Cross-spectrum is identified by the ‘\_specx’ suffix and has a dimension of [a x 19]. When a spectrum variable is highlighted in the **New Variables** listbox, users can read the cross-spectral data, including auto-, co-, quad-spectra, transfer function, coherence, etc. by clicking the **Read** button. Users can read only a specified period range by changing the values in the **MinMax Pd** textbox. If the **Band Average** checkbox is on, the displayed spectra is band-averaged based on the default averaging scheme: [1 40; 2 5; 5 6; 10 30; 20 15; 50 6; 100 30; 200 15; 500 6]. That is, the first 40 bands are not averaged, the next 5 bands are 2-point averaged, the next 6 bands are 5-point averaged, etc. After the **Read** button is clicked, users can also click the **Hardcopy** button to load the spectra data into MATLAB's default text editor. Users need to change the font size to 7 in order to fit the data into the width of a Letter size paper. The font size can be changed from the **Edit** menu (Edit > format...) of the text editor. After the font is changed, the data can be sent to the printer.

Auto-spectra can also be estimated from ADCP variables. Its dimension is in a form of [a x 5 x b] where b is the number of bins. Cross-spectrum for ADCP variables are not available, but under development.

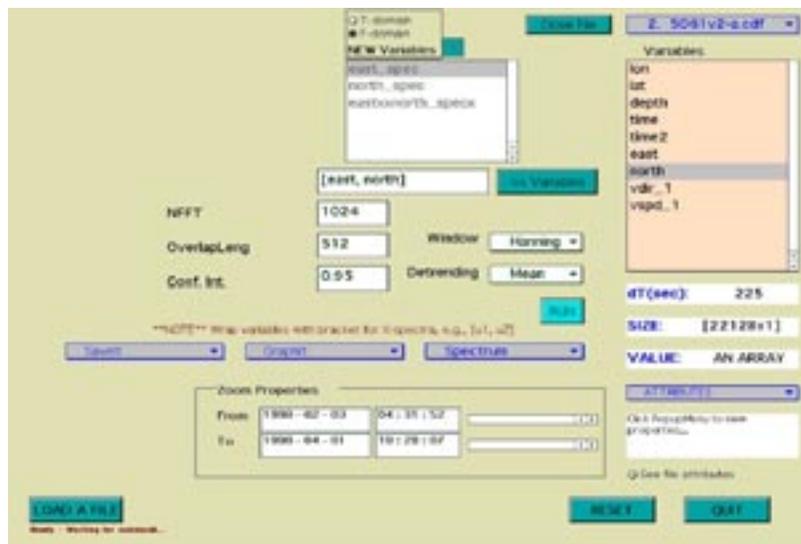


Figure 11. The screen shot of the CMGTool GUI window in which the cross-spectrum of two variables (east, north) is to be computed when users click the **RUN** button.

### 3.5.2 GraphIt – graphic presentation of spectra

GraphIt is used to plot spectral variables. Users may choose from **LogLog**, **SemiLogx**, **SemiLogy**, **Linear**, or **Variance-preserving** plots. The last plot type can only be used on auto-spectra only. Users also have the option of plotting band-averaged or unaveraged spectra. The

frequency range of the plot may be changed. When plotting a cross-spectra, the auto-spectra, clockwise/anticlockwise spectra, phase, and coherence are plotted in 4 different subplots (Figure 12). When plotting auto-spectra of ADCP data, the spectra are displayed as a 3-D waterfall plot if the spectra are from more than two ADCP bins (Figure 13). Users can choose which bin(s) are to be plotted.

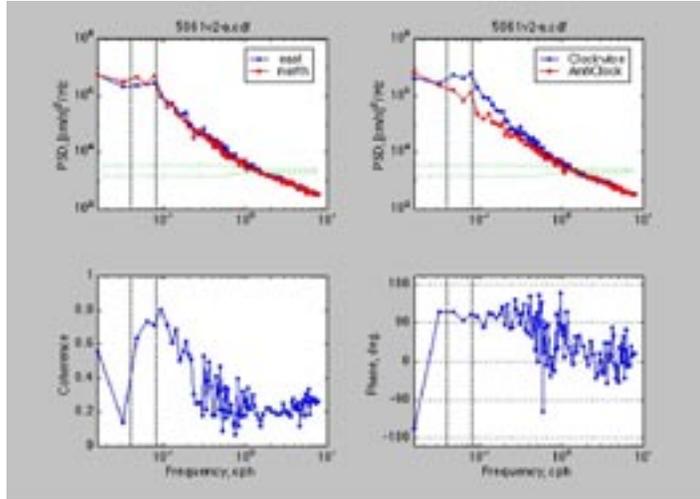


Figure 12. Example plot of a cross-spectrum. The auto spectra, rotary spectra, phase and coherence are plotted in 4 separate subplots.

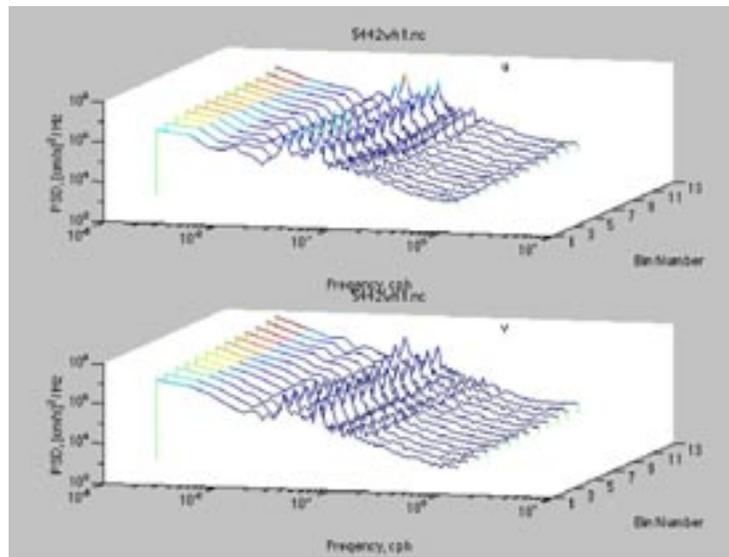


Figure 13. The waterfall plots of the auto spectra computed from an ADCP time-series data.

### 3.5.3 SaveIt – Saving variables into new files

Spectral data can only be saved into ASCII or MATLAB format. NetCDF format has not been implemented in the current version. See Figure 10 for instructions of putting variables into the **Save Variables** listbox.

## 4. CMGTool LIBRARY

The CMGTool GUI is designed to provide a user-friendly analyzing tool of oceanographic data to the users who would like to utilize the powerful MATLAB processing capability, but might not be proficient in the MATLAB language itself. A shortcoming of the GUI is that it cannot be run in a batch mode. The CMGTool library was therefore developed to not only enable batch processing of most of the GUI functions, but also to provide a basic data processing toolbox that experienced MATLAB users, who may prefer 'command-line' computing, can utilize. Below, the library functions are listed alphabetically and briefly described. All the programs here are stand-alone routines, many of them are not included in the CMGTool GUI.

Function: **adr2mat** (PC and UNIX only)

Purpose: Read the binary file from an ADV recorder into a Struct in the workspace.

Syntax: `[xyz] = adr2mat (adrname, [btORbk])`

Input: *adrname*: the binary file name, a string;

*btORbk* -when it is bk (blocksize): An integer value in MB, optional. It allows users to save the struct (*xyz*), read from a LARGE binary file, into multiple mat files when the computer memory is limited.

*btORbk* ≤ 0, No mat file saved, same as *btORbk* being omitted, default.

*btORbk* > 0, divides the binary file into blocks of [*btORbk*] (approximately) MB and saves each block, along with the metadata, into a mat file identified by the ADV serial number. The saved file could be several times larger than *btORbk* MB.

*btORbk* -when it is bt (burst range): In the form of [b0, b1] where b0 and b1 represent the starting and ending burst numbers. It MUST have two values and b1 ≥ b0.

Output: *xyz*: The Struct that holds everything, optional

Function: **adr2nc** (PC and UNIX only)

Purpose: To read a binary file (\*.adr) from an ADV recorder and write the data into a netcdf file. The new file has the same base name but with the '.nc' extension.

Syntax: `adr2nc (adrname)`

Input: *adrname*: a string, the name of binary \*.adr file

Output: none

Function: **advmat2nc** (PC and UNIX only)

Purpose: Creates a NetCDF file from a Struct generated by **adr2mat**

Syntax: `advmat2nc (mystruct, [ncfilename])`

Input: *mystruct*: the struct created using **adr2mat**

*ncfilename*: optional. the netcdf file name (a string).

Output: none

Example: `advmat2nc (xyz, 'southcal.nc')`

or

`advmat2nc(xyz);` which create a netcdf file named 'xyz.nc'

Function: **cmgadcpview**

Purpose: View profile of ADCP data

Syntax: `cmgadcpview (jdaytime, east,north [,dep, angle]);` -To view ADCP data that already in the MATLAB workspace.

or

`cmgadcpview (fname, uname, vname [,dname, angle]);` -To view ADCP data from a NetCDF file. Require NetCDF capability on your system, use `ncdump` to see the names U, V, and depth

Input: `jdaytime` = time in true Julian dates  
`east,north` = the east and north components of ADCP data matrix of size (M x N) where M = length of `jdaytime`  
`dep` = array of water depths where measurements are made, OPTIONAL  
`fname` = a text string of the NetCDF file name  
`uname, vname` = the text strings of the east and north velocity components (make sure they exist in the file).  
`dname` = a text string of the depth, OPTIONAL  
`angle` = rotate this angle (in degrees) before viewing, OPTIONAL

Output: none

Function: **cmgavg**  
Purpose: A points-averaging function  
Syntax: `[gout] = cmgavg (gin, theta)`  
Input: `gin`=input data vector. If `gin` is a matrix, each column of the matrix is treated as a vector.  
`theta`=length of average.  
Output: `gout`=output data.  
Example: `ts2 = cmgavg (ts1, 4);` produces a 4-point average.

Function: **cmgbridge**  
Purpose: To fill gaps in time-series files using Joseph's scheme( a spectral method) after filling short gaps with a linear fit from neighbors' values.  
Syntax: `[dat,[nogap]] = cmgbridge (dat, nlin, nmaxbr, maxngaps)`  
Input: `dat` = data to be bridged. if matrix, bridging performed on each column  
`nlin` = max gap length to be filled linearly, optional (default =2)  
`nmaxtbr` = max gap length to be filled spectrally, optional (default = 48)  
`maxngaps` = max number of gaps to be filled, optional (default = 1000)  
Output: `nogap` = 0 if gaps found, 1 if no gaps, optional;  
`dat` = gap-filled data output

Function: **cmgcdemodulate**  
Purpose: Complex demodulation of a vector or scalar variable  
Syntax: `[newt,ap,pp [,am,pm]] = cmgcdemodulate (pd, pl, t, u, [v])`  
Description: Based on algorithm in "Data analyses methods in physical oceanography" by W.J. Emery and R.E. Thomson, page: 402 - 404  
Input: `pd` = the period (in hours) of interest, maybe an [Mx1] array  
`pl` = segment length  
`t` = time variable (in true Julian dates), [Nx1]  
`u` = time-series of interest, [Nx1]  
`v` = same as u, optional  
Output: `newt` = new time at midpoint of segment, in true Julian date  
`ap, pp` = amplitude and phase of counterclockwise components  
`am, pm` = amplitude and phase of clockwise components, output only when v is present

The ellipse parameters may be computed from the above output

semi-major axis =  $ap + am$   
semi-minor axis =  $ap - am$   
inclination (angle between major and u) =  $0.5*(pp+pm)$

Function: **cmgcolock**

Purpose: Checks for time gap. Gaps in time are rebuilt and gaps in data are filled with NaNs

Syntax:  $[t, data] = \text{cmgcolock}(t, [data], [dt])$

Input:  $t = \text{time}$

$data = \text{data, vector or matrix, optional.}$

$dt = \text{time increment of the input data, optional.}$

Output:  $t = \text{time}$

$data = \text{If gaps are found, the same rows of each column are filled with NaNs}$

Function: **cmgdataclean**

Purpose: Replaces bad data points with NaNs

Syntax:  $[gout] = \text{cmgdataclean}(gin, threshold)$

Input:  $gin = \text{input data vector or matrix}$

$threshold = \text{bad data threshold. If the absolute value of the input data points are greater than is, those points are replace with NaNs. optional (default = 1.0E35)}$

Output:  $gout = \text{output data vector or matrix}$

Function: **cmgeof**

Purpose: Empirical Orthogonal Function analysis

Syntax:  $[ampt, amps, sv, mv] = \text{cmgeof}(x, [,y] [,modes]);$

Description: Based on Noble and Ramp (2000), Deep-Sea Res. II (47), 871-906; and Kundu and Allen (1976), JPO (6) 181-199

Input:  $x = \text{data matrix (M x N), M observations and N variables. Maybe complex, in which case, y is not allowed.}$

$y = \text{OPTIONAL, used in complex EOF, same size as } x$

$modes = \text{OPTIONAL, the first [modes] number of modes are outputted, default=N}$

Output:  $ampt = \text{temporal amplitude of each mode (M x num).}$

$amps = \text{spatial amplitude of each mode (N x num).}$

$sv = \text{site variance in percentage (N x num).}$

$mv = \text{modal variance in percentage (num x 1).}$

If complex EOF, the on-screen print-out of both amplitudes have this format  
AMP\*exp(PHASE\*i), where PHASE is in degrees

Function: **cmgidgaps**

Purpose: Identifies gaps in a time series

Syntax:  $[ngaps, nfirstbad, nlastbad, lgap] = \text{cmgidgaps}(dat, maxgaps)$

Description: Adapted from a fortran program bridge.f

Input:  $dat = \text{input data series. If dat is a matrix, gap search is performed columnwise.}$

$maxgaps = \text{max number of gaps to be filled, optional (default = 1000)}$

Output:  $ngaps = \text{the number of gaps to fill (excludes gaps that can't be filled because they go to the end of the file)}$

$nfirstbad, nlastbad, lgap = \text{integer arrays containing the indices for the first bad point of each gap, the last bad point, and the length of each gap}$

Function: **cmgmat2nc**

Purpose: Converts a MATLAB workspace or MATLAB files into a NetCDF file(s)

Syntax: `cmgmat2nc (ss, [matname], [cdfname])` – Converts some variables in the MATLAB workspace or a MATLAB file to a NetCDF file.

Or

Input: `cmgmat2nc (batchname)` – Converts one or multiple MATLAB files to NetCDF files  
*ss* = a cell array that lists the names of all variables (the first one MUST be the time variable) that will be saved into the new NetCDF file. Strings may be case-sensitive depending on platform. An example of *ss* reads like *ss*={'velU','velV','temperature'}.  
*matname* = optional, the name of the MATLAB file containing all the variables. If omitted, the function will try to load variables from the 'base' workspace.  
*cdfname* = optional, a string designates the NetCDF file name. ONLY used when a *matname* is provided.

Batchname = the name of a text file that contains all information of the source files/variables that will be written into NetCDF files. The format of the text file must be strictly followed. Example batch file:

```
[1] test.mat %mat file name
time, velU, velV, temp %variable names
Jdays, cm/s, cm/s, degC %variable units
0, 25, 25, 25 % depth (M) where measurements were made
GMT %time zone
33 34.6 %latitude
-117 56.5 %longitude
62 %water depth (M)
USGS %data origin
656 %mooring number
OCSD, M07 mooring %description
Noble %PI
13.75EAST %magnetic variation
Time series %data type
Geographical %coordinate system
Data chopped from tail %data comment
ADV %instrument
B206 %serial number
Created using cmgmat2nc %history
Mync.nc %NetCDF file name
[2] test2.mat
.
.
.
[3] test3.mat
.
.
```

Output: none

NOTES: The timebase and all variables listed in *ss* will be loaded into the workspace first. So it may take a few seconds...

Function: **cmgnielsen**

Purpose: Computes local wave number using Nielsen (1982)'s empirical function.

Syntax: `klocal = cmgnielsen (period, depth)`

Input: *period* = vector of wave period in SECONDS  
*depth* = vector of water depth in METERS

Output: *klocal* = output local wave number



*theta* = the angle (degrees) of rotation, positive in counterclockwise

Output: *east* and *north* must be the same size. If matrices, rotation is performed columnwise.  
*newx*, *newy*, *newdata* = data output after rotation.

Function: **cmgsmoothie**

Purpose: Smooths a time-series using MATLAB function conv.m

Syntax: *smoothed* = cmgsmoothie (*vari*, [*fsize*]);

Input: *fsize*: size of a kernel, optional, (default = 3);

*vari*: variable to be smoothed, vector or matrix; if a matrix, convolution performed on each column.

Output: *smoothed*: the smoothed time-series

Function: **cmgspacebuster**

Purpose: Removes both the leading and trailing white spaces from string

Syntax: *newstring* = cmgspacebuster (*oldstring*)

Function: **cmgspd2uv**

Purpose: Creates the East and North components from speed and direction in Cartesian coordinate.

Syntax: [*east*, *north*] = cmgspd2uv (*spd*, *dir*)

Input: *spd* = speed

*dir* = direction (degrees) in true north

Output: *east*, *north* = east and north components vector or matrix

*spd* and *dir* must be the same size. If matrices, calculation is performed columnwise.

Function: **cmgspecavg**

Purpose: Band-averaging spectra, PROSPECT style.

Syntax: *theSpec* = cmgspecavg (*specStruct*, [*nfreq*])

Description: REQUIRE spectral output from **cmgspectra**. Modified from Chuck Denham's *specav.m*

Input: *specStruct* = Structure array from **cmgspectra** that contains these fields:

.spec: auto- (size of M x 5) or cross- (size of M x 19) power spectra

.freq: (M x 1) frequency in Hz

.npieces: number of pieces

.conf: confidence level

.window: no window(=0); hanned (=1)

.nfft: nfft

*nfreq* = optional, a (N x 2) matrix specifying a varying number of frequencies in each band, e.g. a [4 5; 10 15; 28 30] matrix specifies that there will be 4 frequencies in the first 5 band, 10 in the next 15 bands, and 28 in the next 30 bands.

Output: *theSpec*, is also a struct similar to the input *specStruct*.

Function: **cmgspectra**

Purpose: Computes auto- and cross- spectra.

Syntax: *results* = cmgspectra(*x*, [*y*], [*nfft*], [*noverlap*], [*window*], [*fs*], [*p*], [*dflag*])

Description: The output are powers, not power density.

Inputs:  $x$  = time-series, vector or matrix  
 $y$  = second time-series, optional. If present, both auto- and cross- spectra are calculated, and both  $x$  and  $y$  must NOT be a matrix.  
 $nfft$  = length of each section to be FFT'ed (default=256)  
 $noverlap$  = length of overlap (default= $nfft/2$ )  
 $window$  = (default=hanning( $nfft$ ))  
 $fs$  = sampling frequency (default=2)  
 $p$  = confidence interval (default=0.95)  
 $dflag$  = detrending mode (default='linear')

You can obtain a default parameter by leaving it out or inserting an empty matrix [], e.g. `cmgspectra(x,y,[],128,[],2)`, in which  $nfft$  and  $window$  are set to default,  $p$  and  $dflag$  are left out (default).

Output: results = A struct containing the following fields:  
.spec (M x 5 for auto-; M x 19 for cross- spectra)  
.freq  
.npieces --used in averaging in the plot routine  
.nfft  
.conf (a scalar, confidence level)  
.window  
.name  
.averaged

Function: **cmgstats**

Purpose: Computes the error bar, mean, standard deviation, maximum and minimum of a time-series.

Syntax: `cmgstats(x, [dof])`

Input:  $x$  = time-series, vector or matrix. If matrix, calculations are performed along each column.

$dof$  = degree of freedom, optional. If omitted, users are asked to choose a value of autocorrelation scale to estimate the  $dof$ .

Output: All output are printed in the command window.

Function: **cmgtssview**

Purpose: Displays vertical profile of temperature, salinity or density collected from moorings.

Syntax: `cmgtssview(jdaytime, dep, temp);`

Input:  $jdaytime$  = time vector in true Julian days

$temp$  = temperature, salinity or density matrix of size (M x N), where M = length of  $jdaytime$

$depth$  = array of water depths where measurements are made

Output: none

Function: **cmguv2spd**

Purpose: Creates speed and direction from (u, v) in Cartesian coordinate.

Syntax: `[spd, direc] = cmgspd2uv(east, north)`

Input:  $east, north$  = east and north components vector or matrix

Output:  $spd$  = speed

$direc$  = direction (degrees) in true north

$east$  and  $north$  must be the same size. If matrices, calculation is performed columnwise.

Function: **cmgwindstress**

Purpose: Computes wind shear stress based on given speed, direction, and sensor height.

Syntax: `[ustr, vstr] = cmgwindstress (wspd, wdir, [height])`

Input: *wspd* = wind speed vector in m/s

*wdir* = wind direction vector in degrees (true North)

*height* = height of measurement in m above sea level. if omitted, height=10 m.

Output: *ustr, vstr* = wind stress in east and north directions.

Function: **cmgxlabel**

Purpose: To relabel the time axis from Julian day (or datenum) to Gregorian labels.

Syntax: `cmgxlabel (jd, [hand])`

Input: *jd* = True Julian day or datenum

*hand* = graphic handle, optional, default is *gca*

Output: none

Function: **nc2struct**

Purpose: Loads a NetCDF file into MATLAB workspace as a struct

Syntax: `x = nc2struct (ncfilename, [vflag]);`

Input: *ncfilename* = name of the NetCDF file

*vflag* = optional. If exist, variable attributes are loaded

Output: *x* = optional. The struct

All global attributes are stored in the 'metadata' field. The attribute of individual variables are not loaded by default, but users have options to load them into the 'vattrib' field.

## 5. PROSPECT-in-MATLAB

PROSPECT is a legacy software package for spectral analysis of equally-spaced time series. The package is written for the VAX 11/780 computer at WHOI, entirely in FORTRAN 77. PROSPECT-in-MATLAB is the MATLAB adaptation of PROSPECT. It has kept some of the features that are familiar to original PROSPECT users. Many others, such as optional switches in some PROSPECT programs, are deemed unnecessary in the MATLAB computing environment and therefore not included in the adaptation. The new codes run much faster due to the matrix operation capability of MATLAB. All program script files are included in the CMGTool library.

Following the tradition of PROSPECT, a typical PROSPECT-in-MATLAB session may be summarized in flow-charts displayed in Figure 14.

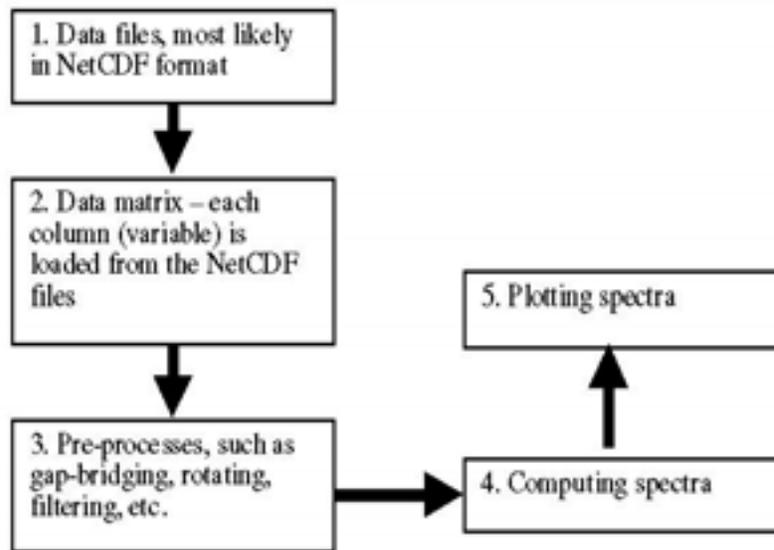


Figure 14. Flow-chart of a typical PROSPECT-in-MATLAB session

### 5.1 Data files

PROSPECT in MATLAB assumes that all data files are in NetCDF format. In order to be able to manipulate NetCDF files, a NetCDF gateway for MATLAB, **mexcdf**, and the **NetCDF toolbox** (see <http://crusty.er.usgs.gov/~cdenham> for details) have to be installed on your computer. All data files must be in the current directory or in a directory that is listed in MATLAB's searching path.

### 5.2 Building the data matrix with **cmgfbuild**:

Function: **cmgfbuild**

Purpose: Reads all specified variables from all relevant files that are also specified in a user-defined text file.

Syntax: *mydata=cmgfbuild ( infoname, [dtflag], [tflag] )*

Input: *infoname*= name of the text file that contains the name of file-building information. See the example:

```

5121-a.cdf, 5123-a.nc, 5125-a.nc, 5126-a.nc, 5127-a.cdf % the names of input files
2, 2, 2, 2 % the number of variables in each file
east, north, u_1205, v_1206, u_1205, v_1206, u_1205, v_1206, east, north % the
name of variables
3600 % delta-t in seconds
20-Nov-1997 09:00:00 % start time
04-May-1998 09:00:00 % end time
  
```

In this example, the program reads 2 variables (each variable may have multiple columns, as in ADCP data) from each of the 5 NetCDF files. If the sampling intervals of a file is different from 3600 seconds, the variables are interpolated so that all variables in the result data matrix have the same delta-t as well as the same start and stop times that are given in the last two lines of the example file.

*dtflag*= optional, when time-series of different dt are loaded, *dtflag* helps determine either the smallest (=0, default) or largest (=1) dt to use to form anew timebase. If dt is provided in the *infilename* file, *dtflag* is not used.

*tflag*= optional, when time-series of different lengths are loaded, *tflag* helps determine either to shrink (=0, default) or expand (=1) in forming a new timebase.

Output:

*mydata*: a structure array that contains the following fields.

.time: time in Julian days

.data: the data matrix (M x N for single column variables, M x L, L>N for ADCP)

.columns: a [1 x N] array containing the number of columns (bins) in each variable

.fname: a [1 x N] cell array providing the name of source files for each variable

.vname: a [1 x N] cell array of variable names

### 5.3 Pre-processing the data matrix, *mydata*, if necessary:

Gap-bridging: *mydata.data*=**cmgbridge**( *mydata.data*, .... )

Filtering: *mydata.data*=**PL33** ( *mydata.data*,.... )

Rotating: *mydata.data*=**cmgrotate**( *mydata.data*,....*a* )

### 5.4 Computing spectral properties using **SPECACR**:

Function: **SPECACR**

Purpose: Spectra estimator

Syntax: *myspec* = **SPECACR** ( *mydata*, *infilename* )

Input: *mydata*: the struct variable generated by **cmgbuild.m**

*infilename*: Name of a file that contains parameters used in spectral computation. An example is listed below (the format must be strictly followed):

```
%%%%%%%%%% information used in SPECACR.M %%%%%%%%%%%
[1,2], [1,3], [2,3] % specify which pairs of variables to compute spectra
[3,4], [2,5], [1,1] % specify which column of each variable is used (for ADCP); for
                    single-column variables, all entries should be 1.
```

```
[ ] % starting point, default is 1
```

```
[ ], % length to use, default is to use entire series
```

```
[ ] % band-avg param, ****SEE BELOW FOR DETAIL****
```

```
1024 % length of each section to be FFT'ed (default=256)
```

```
0 % length of overlap (default=0)
```

```
boxcar % windowing (hanning or boxcar) (default=boxcar)
```

```
3600 % sampling interval in seconds (default=0.5)
```

```
0.95 % confidence interval (default=0.95)
```

```
none % detrending mode (default=none)
```

```
% ****DETAIL OF BAND AVERAGING INPUT****
```

```
% if the entry is [ ]
```

```
%     use the default ([1,40], [2,15], [5,6], [10,30], [20,15], [50,6], [100,30],
[200,15], [500,6])
```

```
%
```

```
% if the entry is 0,
```

```
%     there will be no band averaging
```

```
%
```

```
% if the entry is an scalar > 1,
```

```
%     using the same number of freq in each band
```

```
%
```

```

% if the entry is like [4,5], [10,15], [28,30]...
%     using the actual input
% *****DETAIL OF BAND AVERAGING INPUT*****

```

Output: *myspec*: A struct array, each of the N elements contains the following fields.

- myspec* (1:N).spec (M x 5 for auto-; M x 19 for cross- spectra)
- myspec* (1:N).freq
- myspec* (1:N).npieces --used in averaging in the plot routine
- myspec* (1:N).nfft
- myspec* (1:N).conf (a scalar, confidence level)
- myspec* (1:N).window
- myspec* (1:N).name
- myspec* (1:N).averaged

### 5.5 Generating spectral plots using **SPECPLT**:

Function: **SPECPLT**

Purpose: Creates spectral plot

Syntax: **SPECPLT** ( *myspec*, *fname*, [*options...*] )

Input: *myspec*: see above.

*fname*: base string for filenames of plotted graphics.

*options*: (optional)

one of the following output types:

'jpeg' (default)

'ill'

'epsc'

AND/OR

one of the following plot types:

'loglog' (default)

'semilogx'

'semilogy'

'linear'

'vp' (variance preserving)

AND/OR

one of the following plot contents:

'auto': plotting two auto spectra

'coquad': plotting co- and quad spectra

'chph': plotting coherence and phase

'rotary': plotting rotary spectra

'combo': plotting auto-, rotary, coherence, and phase in one plot (default)

Output: There is no output into the MATLAB workspace. Each and every appears briefly on screen before being overlaid by the next plot. All plots are saved to files whose name are specified by the graphic format, plot content, and pair number. These files are stored in the current directory.

## 6. ACKNOWLEDGEMENT

The authors would like to thank Rich Signell, Chris Sherwood, and Bob Beardsley for sharing their elegant codes. We also appreciate all the suggestions and program testing efforts from Marinna Martini, Holly Ryan and Leslie Rosenfeld.

## 7. REFERENCE

- Aquatec (2001) Acoustic Back Scatter (ABS) System. Aquatek Electronics Limited, United Kindom, p.2.
- Emery, W.J. and R.E. Thomson (1997) Data analysis methods in physical oceanography. Elsevier Science Inc., New York, p634
- Forman M.G.G. (1978) Manual for tidal currents analysis and prediction. Pacific Marine Science Report 78-6, Institute of Ocean Sciences, Sidney, British Columbia, p.57.
- Nielsen, P. (1983) Analytical determination of nearshore wave height variation due to refraction, shoaling and friction. Coastal Engineering, 7:233-251.
- Noble M.A. and S.R. Ramp (2000) Subtidal currents over the central California slope: evidence for offshore veering of the undercurrent and for direct, wind-driven slope currents. Deep-Sea Research, Part II, 47(5/6) 871-906.
- Soreide, N. (1997) EPIC - A system for management and analysis of oceanographic time series and hydrographic data. Pacific Marine Environmental Laboratory, NOAA, Seattle, WA, p.64
- RDI (1996) Acoustic Doppler Current Profiler - Principles of operation: A practical primer. RDI Instruments, San Diego, CA., p.54
- Rew, R., D. Glenn, S. Emmerson, and H. Davies (1997) NetCDF User's Guide for C. University Corporation for Atmospheric Research, Boulder, CO., p149.
- SonTek (2001) SonTek/YSI ADVField/Hydra operation manual. SonTek/YSI, San Diego, CA., p121.
- The Mathworks Inc. (1997) MATLAB-The language of technical computing. The Mathworks, Inc., Natick, MA.
- Wu, J. (1980) Wind-stress coefficients over the sea surface near neutral conditions. Journal of Physical Oceanography, 10:727-740